Optical Splitting Trees for High-Precision Monocular Imaging

Morgan McGuire
Williams College

Wojciech Matusik and Hanspeter Pfister
Mitsubishi Electric Research Laboratories (MERL)

Billy Chen
Stanford University

John F. Hughes
Brown University

Shree K. Nayar
Columbia University

The authors present a design framework and optimizer for computational systems containing many sensors on a common optical axis.

Many computational photography applications require sets of images that are captured simultaneously from the same viewpoint but have different image sensors and imaging parameters. In this article, we address the problem of designing efficient multisensor 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 contains as many as eight sensors. (To gain a better sense of the issues we're tackling in this article, read the "Basic Background and Considerations" sidebar.)

In this article, we consider the design of monocular multiview optical systems that form optical splitting trees, where the optical path topology takes the shape of a tree because of recursive beam splitting. Designing optical splitting trees is challenging when it requires many views with specific spectral properties. We introduce a manual design paradigm for optical splitting trees and a computer-assisted design tool 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, measurement accuracy, and economy. An optimizer then searches for a design that maximizes these weighted priorities. Our tool's 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're superior.

We've previously reviewed and demonstrated capturing multiple monocular views in various contexts, but rarely have more than three simultaneous views been captured. (To see others' approaches to this and similar problems, see the "Related Work" sidebar on page 40.) Our approach makes design, implementation, and calibration practical for many more sensors. To demonstrate the practical application of our splitting tree paradigm and optimizer, we built the configurable optical splitting tree system shown in Figure 1 (on page 34) that captures up to eight views. Assisted by our design tool, we were able to use this single device to implement several different popular computational videography applications: high-dynamic range (HDR), multifocus, high-speed, and hybrid high-speed multispectral video. In this article, we demonstrate results for each of these.

Optical splitting trees

Optical splitting trees are schematic representations of 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, a sensor's tree depth (the number of internal nodes between it and the root) might 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. In each case in Figure 3, light enters at the top and emerges at the bottom. Spectral filters are narrow bandpass filters centered at the specified value. "Sensor + Lens" sets specify imaging properties. The neutral density (ND) filter reduces the amount of transmitted light, equally across all wavelengths. The plate mirror splits incoming light into two directions. Unless specified, it reflects 50 percent and transmits 50 percent of

Published by the IEEE Computer Society

Basic Background and Considerations

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. We consider the output computational because the final image isn't (and can't be) recorded by an isolated image sensor; a battery of sensors and computations produce it. Other popular applications that require multiple images and computations include high-speed, super-resolution, focus/defocus analysis, and multispectral video.

We say that the image sets contain multiple monocular scene views 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 correspondence. Except for calibration, they don't need to be warped and don't exhibit parallax artifacts with depth.

The cameras that capture multiple monocular views differ from conventional cameras. A conventional camera contains a single objective (lens system) and imager. To capture multiple views, the camera makes an optical copy of the incident light using a beam splitter, such as a half-silvered mirror.

This lets two imagers 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 imager. As with the HDR example, we can combine these measurements into an image that represents more information (bits) than a single view, allowing for high-precision measurements. The method of combining the measurements may increase the dynamic range (high-order bits), the resolution (low-order bits), the number of samples acquired at different parameters, or some combination.

The amount of useful precisions for a measurement is limited by environmental and sensor noise. The ratio of measured light to incident light is that optical sensor's efficiency. Efficiency and the intensity of the light source determine the amount of noise in an optical measurement. Therefore it's good to design optical systems where a large fraction of light reflected off the subject actually reaches the sensor instead of, for example, being absorbed by filters or lost at lenses through absorption and internal refraction.

When considering a system's efficiency, it's useful to discuss each component's efficiency—for example, a lens that absorbs 10 percent of the incident light is only 90 percent efficient. Although overall efficiency is desirable, sometimes someone intentionally uses an inefficient component. A common example is a neutral density filter, which modulates overly intense light. Considering efficiency over the whole spectrum, a band-pass filter is also intentionally inefficient because it blocks light outside the pass band.

An accurate sensor measures the desired quantity. This is a separate concept from precision. For example, a bathroom scale with five decimal places and a brick on top of it measures weight with high precision but low accuracy because of the brick's added weight. In contrast, a well-calibrated scale with no decimal places has high accuracy but low precision. To make an optical design accurate we must match the real-world specifications of components like filters and sensors to the theoretically desired ones and calibrate for common sources of error like element misalignment and radial distortion. Of course, components with tightly controlled specifications could be prohibitively expensive—economy is often a real limiting factor for both products and research. Design is the process of balancing theoretical desires, engineering capability, and the four concerns of efficiency, accuracy, precision, and economy.

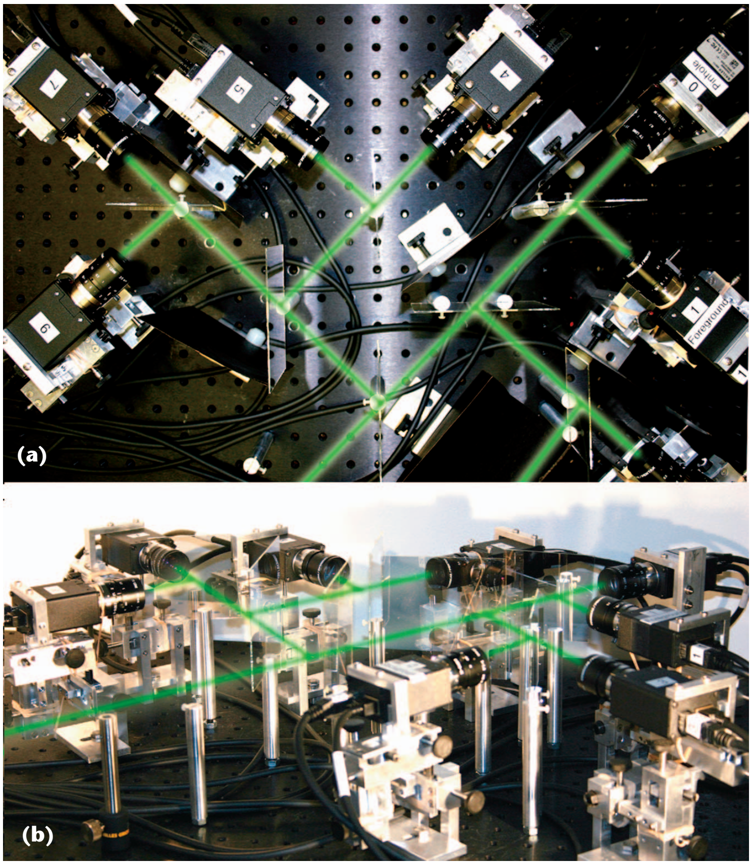
incident light ("50R/50T"). Dichroic mirrors reflect or transmit light according to the incident wavelength.We know that all paths from the root to the leaves have the same physical length, so the representation need not preserve distances. Angles are an artifact of building the physical system and also need not be represented.

We're 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 plate mirrors (as in our implementation). Other splitting elements can produce higher-degree nodes. We can collapse subtrees of beam splitters with no intervening filters into a single node representation with many children, as in Figure 4 (on page 35). This representation also abstracts the nature of the employed splitting element .

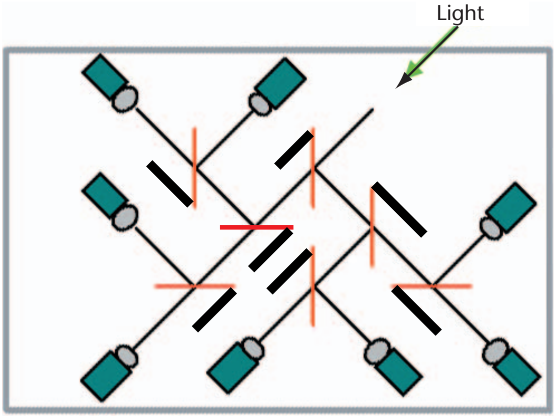
Figure 5a shows a balanced multispectral splitting tree. Each beam splitter is a dichroic mirror, which divides half the remaining visible spectrum among its children so that no light is lost. This is an excellent approach, except that manufacturing large dichroic mirrors is expensive. Figure 5b schematically shows a less efficient design using readily available bandpass filters (for example, Edmund Optics' filters NT43-055 through NT43-088). Each path receives about 1/8 of the incident light, which is bandpass 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 (such as HDR) it's 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.

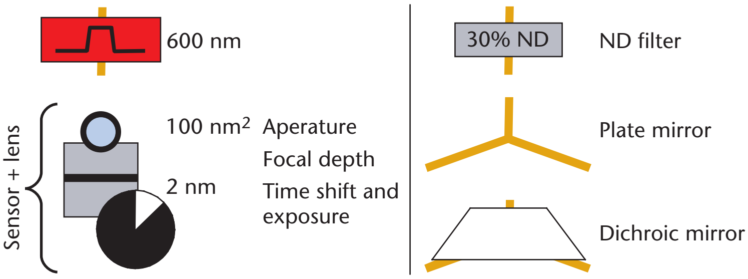
The two designs that we present each have their advantages. Both designs are straightforward ones that someone could construct manually. Other applications offer more alternatives (for example, Figures 5 and 6 demonstrate cost and efficiency tradeoffs), or have spec-



1 (a) Top and (b) side images of our generic eight-view splitting tree system, which can quickly be reconfigured to implement previous capture systems such as high-dynamic range (HDR) and multispectral video, or to meet novel applications. The superimposed laser beam shows the recursively split optical path.



2 Physical layout of a balanced eight-view splitting tree. The red lines are beam splitters and thick black lines are baffles.



3 Lexicon of symbols used in our abstract tree diagrams.

ifications that force tough compromises between desirable characteristics.

Consider a hypothetical application that requires HDR in the red and near-infrared channels, moderate dynamic range across the visible spectrum, and a final catch-all with low-precision, wide-spectrum measurement (we could easily envision face recognition applications with similar requirements). Furthermore, assume that the ideal bit depths of the different channels are all specified by the theory behind this application. Many splitting trees exist that meet these specifications. This immediately raises nontrivial design questions: Should one split the color channels out first? Should the red channel of the visible spectrum sample be used in the HDR red/infrared, even though it only overlaps partly with the desired spectral response? Should exposure, aperture, or neutral density (ND) filters be used to create the HDR trees? Which brand of filter produces the least light loss?

Like most graphics and vision researchers, we've previously answered questions like these by evaluating a handful of promising straw-man designs and ignoring the full design space of possibilities. If the tradeoffs are heavily biased (for example, if the budget isn't a major factor, but the signal-to-noise ratio is crucial), it's not surprising that we can resolve many design decisions by intuition. In those cases, we make reasonable ad-hoc decisions.

What is surprising (at least, it was to us) is that many cases exist where intuitive decisions for seemingly simple applications often produce inaccurate cameras. This can be seen by analysis of the splitting tree for error sources. Figure 7 reveals the primary reason for this. The curves in the figure show the actual spectral response, according to the manufacturers, for nominally uniform components like sensors, plate mirrors, and filters. Ideally, these curves should all be horizontal lines. In reality, they're bumpy curves. This means that a simple design decision like splitting the optical path has unintentionally changed the spectral response and relative efficiency at each sensor.

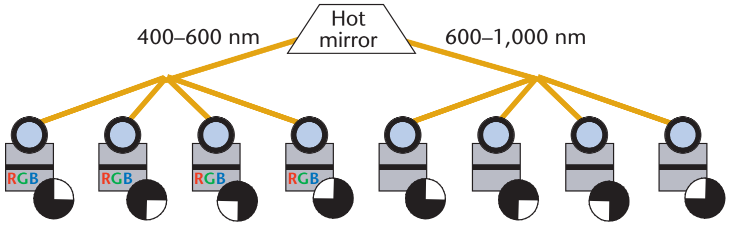
When several splits, filters, and lenses are combined, these deviations from the theoretically ideal behavior create significant inaccuracy. This inherent spectral inaccuracy leads to problems in other parameters. For example, if the red channel is too dim because the beam splitter modulated it unevenly, we must adjust either exposure or aperture to compensate. Alternatively, we could try a different brand of sensor, splitter, filter, or lens. But which of several brands should we use? Even when budgets are flexible, the cost of purchasing precisely uniform components can quickly skyrocket because of manufacturing challenges, so most cameras are designed with nonuniform components.

The error introduced by the deviation between abstract ideal component behavior and real-world specifications motivates a desire to include the real component specifications in the design process. Considering lenses, splitters, and filters, a tree with eight cameras contains dozens of components, some of which are shared over multiple sensors. Adjusting the components along one path affects other downstream sensors, creating a combinatorial configuration problem that then multiplies the difficulty of the previously intuitive design decisions.

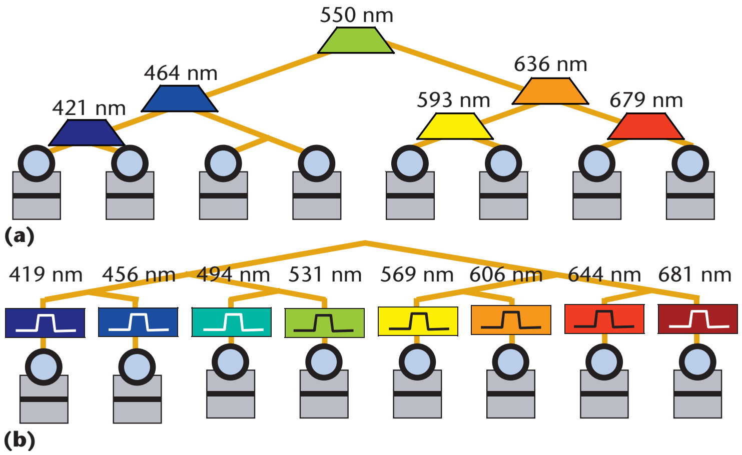
To solve this larger design problem, we created an optimizer that searches design space. Where a human being's design is created under the false assumption that components have ideal properties, the optimizer makes no such assumption and achieves a more accurate fit to the desired specification of the camera as a whole.

Assisted design

We now describe a method for an assisted design tool that creates splitting trees to meet a formal specification. It addresses the complexity of designing large trees and can balance nonideal components against each other. It also takes into account efficiency, cost, and accuracy. The design process contains two phases. The first deterministically constructs an inefficient, expensive tree that's 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 the user explicitly terminates it. Our implementation is in Matlab, using trees from the Data Structures toolbox. We computed all the results in less than an hour, although usually the first minute brought the objective function within 10 percent of the peak value and the optimizer spent the remaining time addressing minor spectral deviations.



4 Splitting tree representation of a hybrid, high-speed, multimodal camera capable of both infrared and visible light capture.



5 (a) Efficient and (b) less efficient bandpass dichroic mirror multispectral trees.

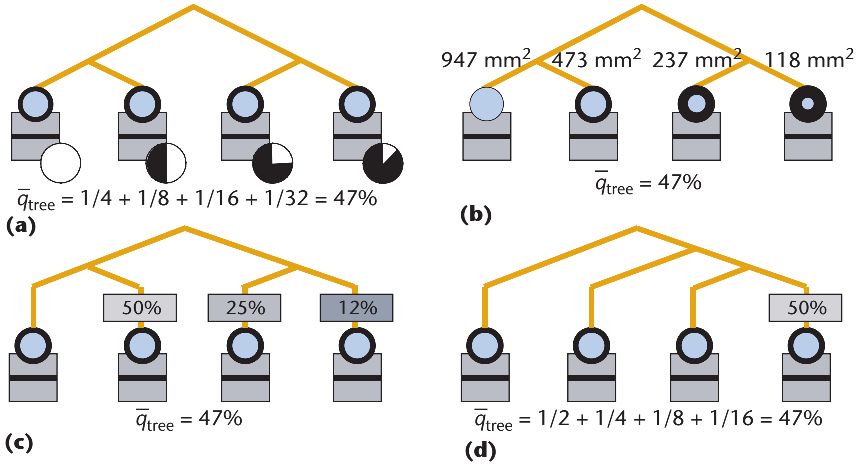
Input specification

Our algorithm takes a set of design specifications $\{\hat{\mathbf{X}}, \hat{\mathbf{I}}, \hat{\mathbf{Y}}\}$ and objective function weights ($\alpha, \beta, \delta, \varepsilon, \gamma$) as input. It also contains a database of components from a real optics catalog (the *Edmund Optics Catalog*—see http://www. edmundoptics.com). We discuss the design specification in the following paragraphs; we discuss the weights in the "Objective function" section.

For each view v (that is, sensor number v), the user specifies the following:

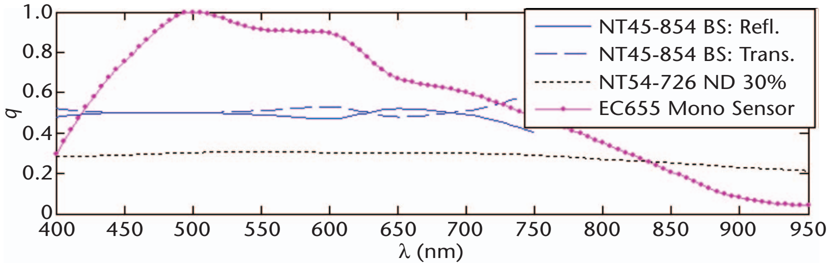
- $\hat{\mathbf{X}}_v[\lambda]$ is the desired spectral response curve, as a unitless vector indexed by wavelengths;
- $\hat{\mathbf{I}}_v$ is the (unitless, scalar) importance of this sensor relative to others.
- $\hat{\mathbf{Y}}_v[p]$ is a vector of the scalar imaging parameters (p): aperture area,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

In this notation, a hat over the variable distinguishes a specification variable from its counterpart that's adjusted during optimization. The subscript v denotes the sensor that a variable describes. Note that no tree structure hints are provided with the specification; the tool begins from scratch.



$\bar{q}_{\text{tree}}$ = 1/4 + 1/8 + 1/16 + 1/32 = 47%
(a)
947 mm^2
473 mm^2
237 mm^2
118 mm^2
$\bar{q}_{\text{tree}}$ = 47%
(b)
50%
25%
12%
$\bar{q}_{\text{tree}}$ = 47%
(c)
50%
$\bar{q}_{\text{tree}}$ = 1/2 + 1/4 + 1/8 + 1/16 = 47%
(d)

6 Some HDR video trees that vary (a) exposure, (b) aperture, (c) filters, and (d) structure. The most efficient approach is (d).



7 Actual efficiency curves for nominally uniform components: beam-splitter #854, neutral-density filter #726, and a Prosilica charge-coupled device. 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.

Efficiency

Efficiency is a desirable quality in a camera. In this section, we rigorously define the efficiency $\mathbf{q}$ of a camera as the fraction of incident light that's measured by sensors.

The specification $\{\hat{\mathbf{X}}, \hat{\mathbf{I}}, \hat{\mathbf{Y}}\}$ 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 7 shows examples of these curves.

We call this curve the quantum efficiency of a component. We represent it with a vector $\mathbf{q}[]$ describing the ratio of output to input light at a node, sampled at 12 wavelengths between 400 and 950 nanometers (nm). Because our components are passive, $\mathbf{q}[\lambda] \le 1$. In other words, the efficiency of a component describes the ratio of light lost as it travels through that component. For a view v, let

$$\mathbf{q}_v[\lambda] = \prod\nolimits_{i \,\in\, \text{path}} \mathbf{q}_i[\lambda]$$

over every component i on the path from v to the root. Let scalar $\bar{\mathbf{q}}_v$ be the mean efficiency of that view with respect to λ, and let

$$\bar{\mathbf{q}}_{\text{tree}} = \sum\nolimits_{v \,\in\, \text{views}} \bar{\mathbf{q}}_v$$

denote the efficiency of the entire camera. With these definitions, $\hat{\mathbf{X}}_v[\lambda] = \mathbf{q}_v[\lambda] / \bar{\mathbf{q}}_v$ specifies the shape of the desired response and $\hat{\mathbf{I}}_v[\lambda] = \bar{\mathbf{q}}_v / \bar{\mathbf{q}}_{\text{tree}}$ is the scalar fraction of measured light captured by view v.

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:

$$
\begin{aligned}
obj\,(\text{tree}) = \quad & \alpha \bar{\mathbf{q}}_{\text{tree}} && (1)\\
& -\sum\nolimits_{\text{views}} (|\beta\,(X_v - \hat{\mathbf{X}}_v)|^2 && (2)\\
& \qquad + |\delta\,(I_v - \hat{\mathbf{I}}_v)|^2 && (3)\\
& \qquad + |\vec{\varepsilon}\,(Y_v - \hat{\mathbf{Y}}_v)|^2) && (4)\\
& -\gamma \sum\nolimits_{\text{nodes}} c_i^2 && (5)
\end{aligned}
$$

These expressions drive the optimizer as follows.

- **Efficiency (Equation 1).** Maximize the total light measured (and therefore, the signal-to-noise ratio).
- **Spectral accuracy (Equation 2).** Minimize the difference from the color specification.
- **Relative importance (Equation 3).** Ensure that each sensor receives the correct amount of light relative to other sensors, regardless of overall efficiency (this is particularly important for applications like HDR where exposures vary widely).
- **Parameter accuracy (Equation 4).** Minimize the difference from the specified pixel shift, temporal shift, polarization, motion blur, and defocus.
- **Economy (Equation 5).** Minimize the purchase cost of components.

Each expression is quadratic to create stable maxima. Recall that $\mathbf{X}$ and $\mathbf{Y}$ 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 meets the specification). They're all scalars except for $\vec{\varepsilon}$, which is a vector so that each imaging parameter may be weighted independently.

We distinguish between scalar (that is, exposure time) parameters in $\hat{\mathbf{Y}}$ and the spectral response vector $\hat{\mathbf{X}}$ for two notational reasons. First, most optical components exhibit surprising spectral nonuniformity but are relatively controlled along other parameters (see Figure 7). Representing spectral error explicitly is therefore informative. Second, the optimization weight is the same for each wavelength. We could instead consider the system to be parameterized only by scalars, where the first 12 represent responses at different wavelengths.

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 $\mathbf{Y}$.) The result quality is most sensitive 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.

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 $\hat{\mathbf{X}}$ to increase the later likelihood of shared filters and then links those trees into a single large tree.

All splitters are 50R/50T plate mirrors at this stage. To satisfy the $\hat{\mathbf{X}}$ values, the system evaluates the actual $\mathbf{X}$ at each leaf and introduces bandpass filters immediately before the sensors to optimize the spectral accuracy term. It then sets all parameters as dictated by $\hat{\mathbf{Y}}$. Finally, it evaluates $\mathbf{I}$ at each leaf and inserts ND filters until the the system satisfies the importance accuracy constraint.

Search phase

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

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

- no transformation (allows $\mathbf{Y}$ change without a tree change);
- if the same filter appears on both children of a beam splitter, move it to the parent of the splitter;
- replace a chain of filters with a single, equivalent filter;
- reorder the filters in a chain (tends to encourage the second and third items);
- rotate a node, as in Figure 8; and
- replace a splitter and filters on its children with a splitter whose R/T ratio approximates the filter ratio

For example, the right rotation transformation in Figure 8 reparents node B. After transformation, the optimizer explicitly adjusts the ND filters to maintain the light ratios at q_A:q_B:q_C = 2:1:1. For the left tree, the fractions of light at the root are 1/4, 1/8, and 1/8. For the right, they're 1/2, 1/4, and 1/4. Both designs capture the same (relative) measurement, but the right tree is twice as efficient. However, spectral filters are expensive and ND filters have negligible cost. The tree on the left is therefore almost twice as economical as the tree on the right.

Transformations other than the ones that we've noted can change $\mathbf{X}$ and $\mathbf{I}$ and are therefore less likely to improve the tree, since the design search begins with a tree that is nearly optimal for those variables. Nonetheless, such transformations provide reachability to the entire design space and therefore allow the optimizer to theoretically find the best tree given sufficient run time. These transformations include

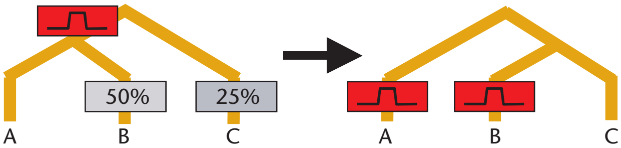
- adding, removing, or changing a filter at random;
- rotating a subtree right or left without adding filters;
- replacing a beam splitter at random; and
- swapping two subtrees.

Experimental system

To test our design framework we built a physical configurable splitting-tree system with eight computer vision sensors that use 100 × 100-mm plate-mirror and hot-mirror beam splitters (see Figure 1). We removed the black baffles in Figure 1a for Figure 1b to make the cameras and light beam more visible from the side. Hot mirrors transmit visible light and reflect infared.

The sensors are Bayer filter A601fc color cameras and monochrome A601f cameras by Basler. Each sensor is equipped with a Pentax objective. The tree exactly fits on a 2 × 2 square-foot optical breadboard with holes spaced at half-inch intervals. The sensors are connected to a single 3-GHz Pentium 4 computer using the FireWire interface. We wired the hardware shutter trigger pins to each of the eight data pins of the PC parallel port, which we precisely controlled by writing bit masks to that port through a special Win32 driver (see http://www.logix4u.net/inpout32.htm).

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 aren't expected to align perfectly.



8 This "right rotation" transformation at the root reparents node B.

To calibrate the system, we first orient the plate mirror beam splitters at 45 degrees 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 bullseyes 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 liquid-eye display (LED) light throughout the scene. Let C_s be the $N \times 3$ matrix whose rows are homogeneous 2D positions—that is, $[x\, y\, 1]$—of the light centroid at subsequent frames in view number v. The transformation mapping pixels in view 1 to those in view v is

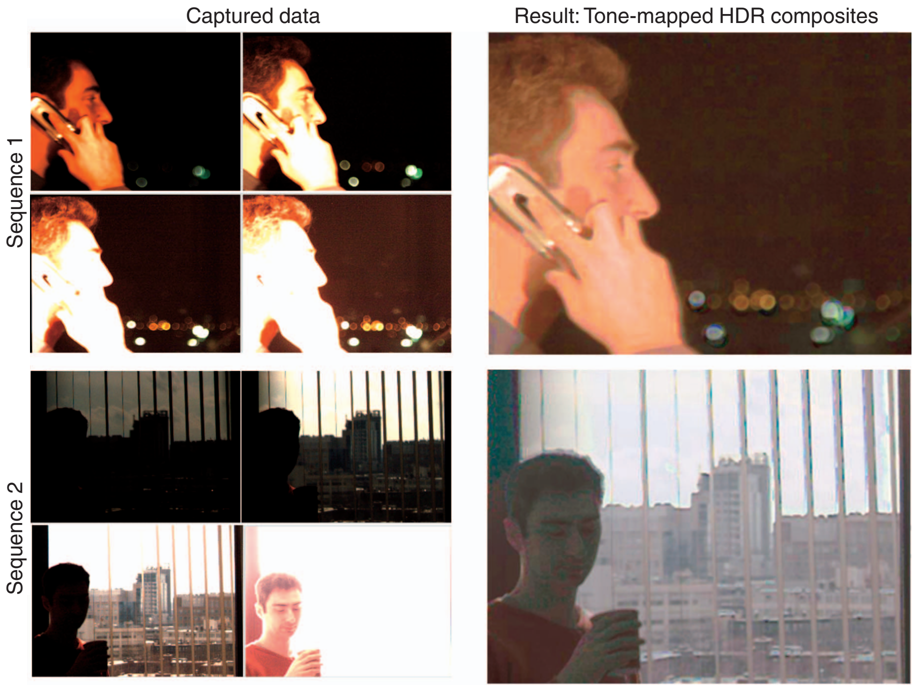
$$H*_v = \arg\min(|H_v C_v - 1|^2) = C_1 C_v^{\dagger}$$

where † 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.

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 that the optimizer automatically produced. Without our system, it often takes several days to assemble and calibrate a new computational photography camera for a single application when starting from scratch. Because our hardware system is configurable and most splitting trees form subsets of the full tree that we prebuild on the optical table, we can configure for different applications comparatively quickly. For each of the examples described here we reconfigured and calibrated 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 cap-



9 **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.**

ture, and registration across many more monocular views than in previous work.

The deterministic phase always produces a viable design, so optimization will technically never fail. However, in about two out of 10 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 nonuniform components, or tricky spectral constraints.

High dynamic range

Figure 6 shows several splitting trees for a simple, HDR 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 Figure 6a, a large economy weight γ gives the inexpensive variable exposure solution.[1] The drawbacks of that approach are inconsistent motion blur between cameras and low $\bar{\mathbf{q}}_{\text{tree}} = 15/32$ efficiency. Increasing the exposure weight and decreasing the aperture weight in $\vec{\varepsilon}$ leads to Figure 6b, where the aperture varies. Now motion blur is correct but the depth of field varies and is still low.

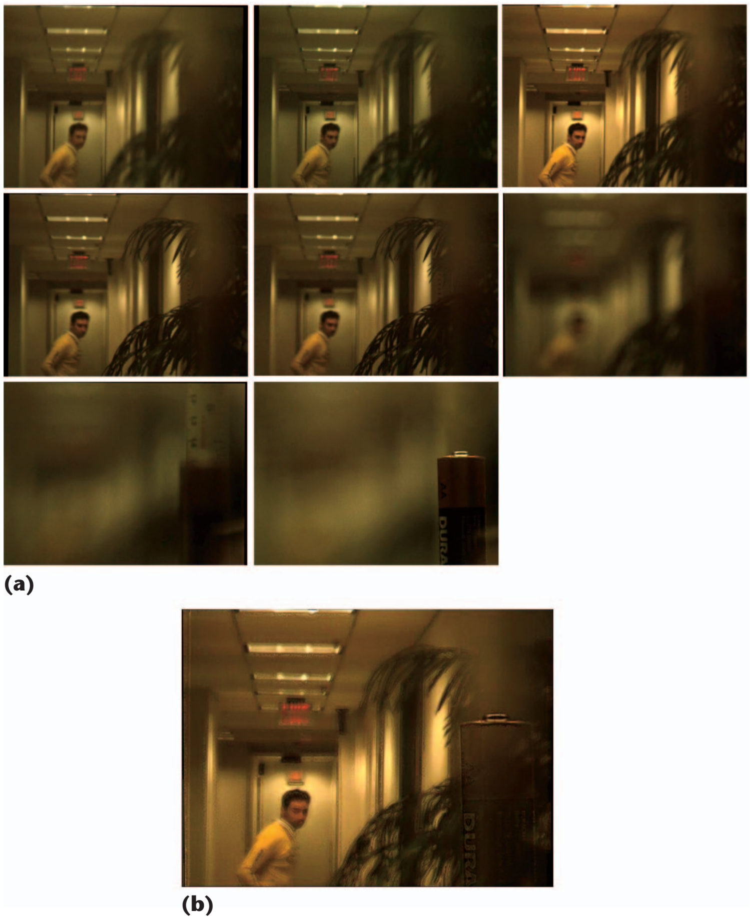
An alternative is to use ND filters as in Figure 6c, similar to Mitsunaga et al.[2] Compared to Debevec and Malik's[1] method, this corrects both motion blur and depth of field but not efficiency. Our optimizer did not rediscover this approach—instead it found a better design!

The tree in Figure 6d has $\bar{\mathbf{q}}_{\text{tree}} = 30/32$. 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[3] mention a similar design in passing; however, their design was never implemented and we believe that it would have produced undesirable asymmetric point-spread functions if actually built.

Figure 9 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 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 display's dynamic range.

Multiple focus and defocus

We can use images focused at multiple depths to recover depth information[4-6] and form images with an infinite[7] depth of field. Many systems[4] split the view behind the lens. Splitting in front of the lens lets us vary not only the location but also the depth of the field by changing the aperture.



10 **Multifocus images captured by sensors. (a) Eight simultaneous views with different focus depths, and (b) a fused infinite depth-of-field result.**

To capture images with varying depths of field, such as those in Figure 10, we use a full binary tree with with eight cameras. Each sensor is focused at a different depth, ranging from 20 cm from the optical center (about 4 cm from the first beam splitter) to 20 m (effectively infinity). We use wide $f/1.4$ apertures for a narrow depth of field on each sensor.

Figure 10 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.

Matting and other designs

Two matting algorithms[8,9] from 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.[8] capture three monocular views, each with a focal length of $f = 50$ mm and each at equal importance: a pinhole view with an aperture, and two views focused at different depths. Their design uses two 50R/50T beam splitters and two filters.

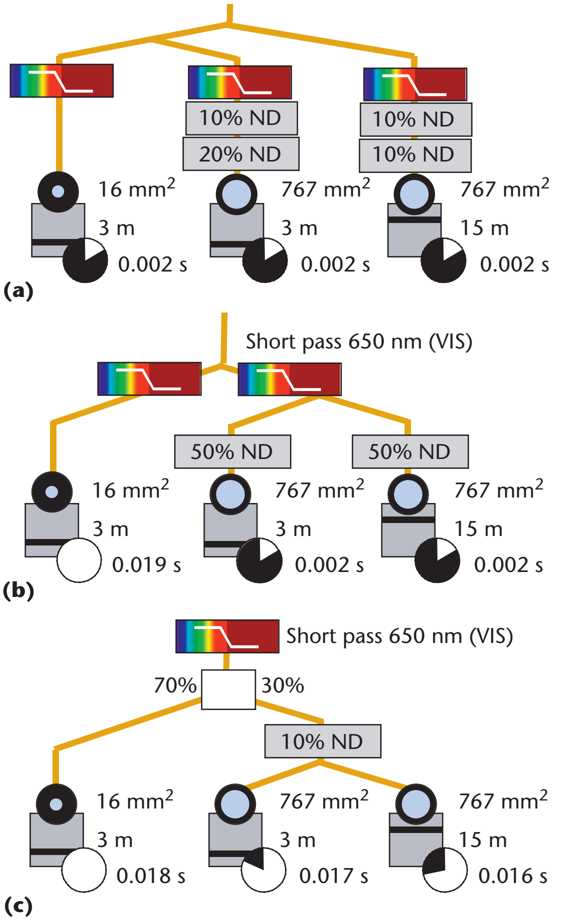
Given their specifications, the deterministic phase of our optimizer immediately produced the correct (but inefficient) tree in Figure 11a. The deterministic phase is constrained to create a full binary tree, place sensors at leaves from left to right, and then add filters at the leaves to implement the desired specification.

After many iterations the optimizer found a better tree shown in Figure 11b. In this tree, the second split is on the right, between the two wide-aperture sensors. This gives more light to the pinhole sensor and reduces the number of ND filters required. Note that the short-pass filter has also propagated up the tree but not yet reached the root.

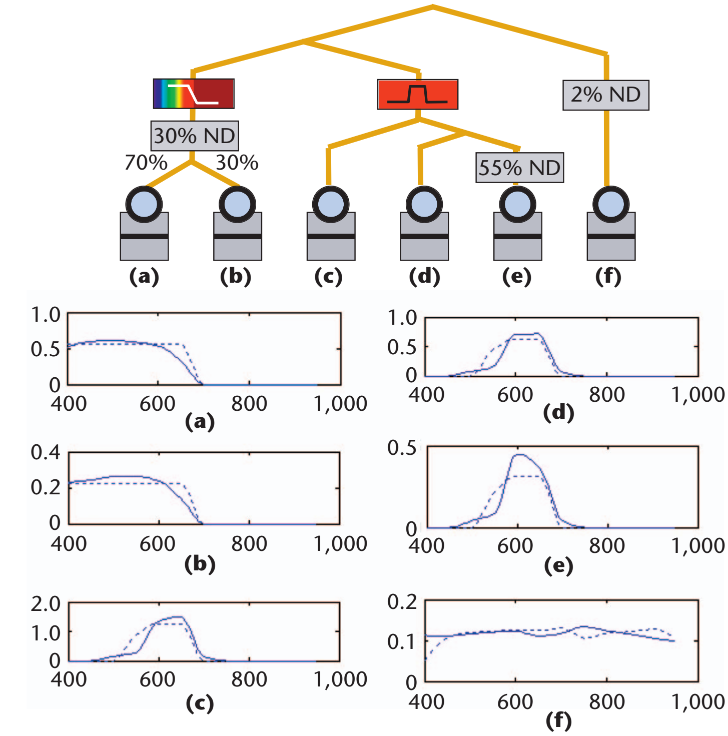
After about 40 minutes, our optimizer appeared to stabilize on the tree design shown in Figure 11c. Compared to the originally published design and trees found on previous iterations, this new design achieves higher efficiency through a 70R/30T beam splitter; a single, weaker ND filter shared across two views to reduce cost; and a short-pass filter to attenuate the color sensors' undesirable infrared response. The optimizer adjusted exposures slightly to compensate for imperfections at the 50R/50T beam splitter on the right branch, where it chose a low-quality component to reduce cost. Subsequent runs produced approximately the same output. Adjusting the input weights traded accuracy in the exposure time against accuracy of the aperture size.

We performed a similar experiment on Debevec et al.'s[9] matting system that uses a plate mirror and an infrared 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



11 **Matting camera based on McGuire et al.[8] at different stages in the search process. (a) After determinstic phase, (b) intermediate result (approximately 30 seconds), and (c) best tree found (approximately 40 minutes). The final camera in (c) is superior to the originally published manual design.**



12 **Plots of (l · x) versus λ nanometers for a complex camera. Dashed lines are the specification, solid lines are the final design; for a good design, these are close in both magnitude (l) and shape (x). The design tool independently matched each spectral efficiency curve close to the specification, taking into account imperfections in the actual filters.**

Related Work

We build on the work of previous authors who have split light paths using a variety of optical methods to capture similar images from multiple sensors simultaneously. Here we review the devices that accomplish this splitting (which are nodes in our splitting tree) and discuss how our approach is relevant to their applications.

Beam splitters

Both prisms and plate 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-charge-coupled device 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 high-dynamic range (HDR) imaging.[1]

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.[2] This 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.[3] use a beam-splitter camera with three views for matting. They report that it can be extended to exhaustively sample various sampling parameters by taking the cartesian product of all sampling parameters and adding large numbers of beam splitters and sensors. We extend these ideas with a detailed discussion and implementation of a full, calibrated eight-view system, analysis methods, and sparse parameter sampling that lets us create views with completely independent characteristics. We demonstrate more efficient and economical camera designs for their application in the "Matting and other designs" section in the main article.

Pyramid mirrors

Another interesting way to create multiple copies involves placing a pyramid mirror behind the lens.[4,5] (Placing a mirror pyramid in front of the lens, as discussed in Tan et al.,[6] creates a panoramic field of view that's unrelated to our work.) Placing the mirror behind the lens creates a 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 we can apply it.

It's also nontrivial 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 Aggarwal and Ahuja.[4]

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 the disk 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.

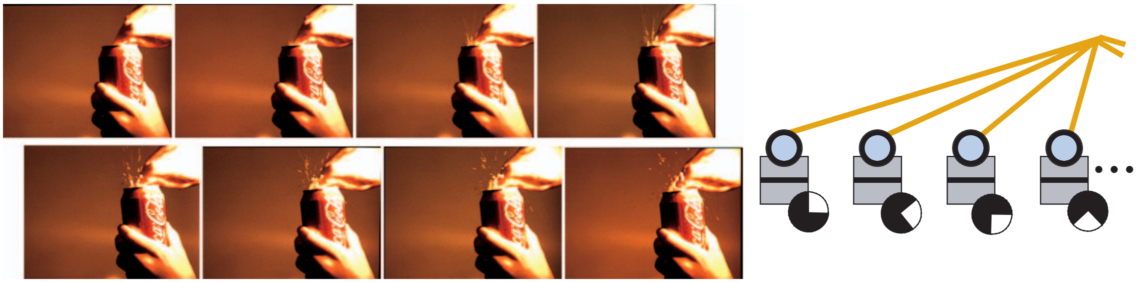
desired accuracy; it even chose between different brands of ND filters based on their q-curves.

High speed

Figure 13 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 1/120th second and the entire sequence is captured at 240 frames per second (fps), so the views overlap temporally. This gives $\bar{\mathbf{q}}_{\text{tree}} = 1/4$, which is lower than the $\bar{\mathbf{q}}_{\text{tree}} = 1$ for an array such as the one by Wilburn et al.[14] The advantage of our approach is that the sensors share an optical center for accurate scene capture with depth variation and view-dependent effects.

Multimodal high speed

High speed, HDR, and so on are sampling strategies. They're useful for building high-level applications such as surveillance in an HDR environment. It's natural to build hybrid sampling strategies, which are easy to express and experiment with in our splitting tree framework.



13 A 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 1/120-second overlapping exposures, longer than is possible for a single-sensor, high-speed camera.

Alternatives

For flat scenes and scenes far (more than10 m) from the camera, parallax and view-dependent effects may be ignored. In that case the calibration problem is comparatively easy, because the sensors' optical centers needn't be aligned. Dense arrays of side-by-side sensors (such as in Wilburn et al.[7]) have captured multiple approximately monocular views for such cases. Arrays capture much more light than a splitting tree. However, a tree can capture both nearby and deep scenes, and can share filters and other optical elements over multiple sensors.

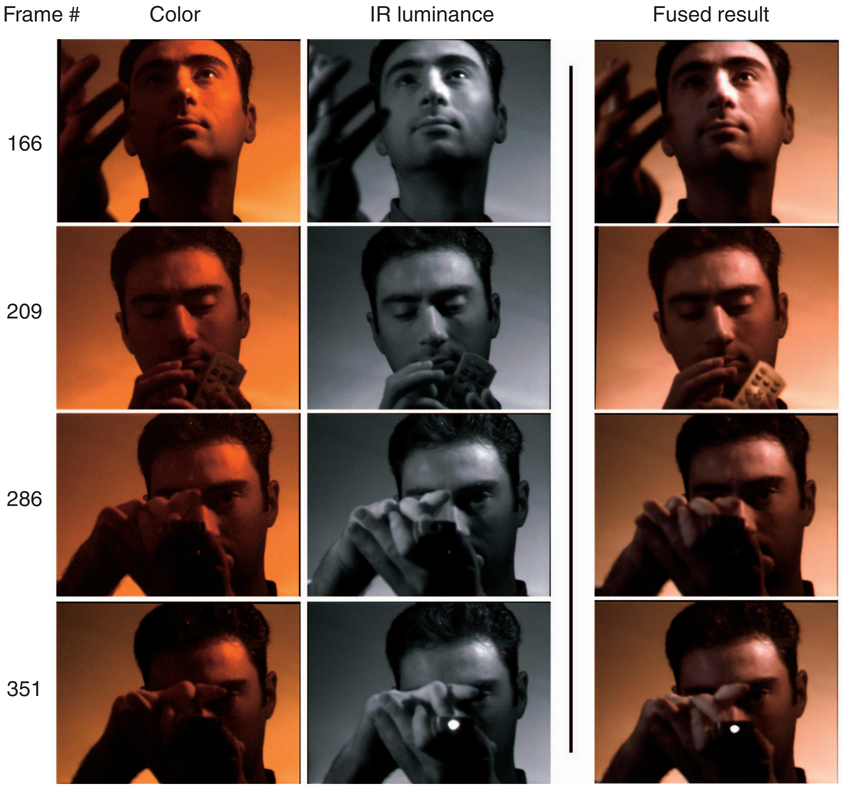
We can use a mosaic of filtered CCD pixels to sample multiple parameters in a single image. The Bayer mosaic tiles per-pixel bandpass filters, sampling three wavelengths with a single monochrome sensor. Recently, some filter mosaics have been proposed for sampling other parameters with high precision.[8,9] 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 we explored in this article.

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 here.

References

1. E. Ikeda, Image Data Processing Apparatus for Processing Combined Image Signals in Order to Extend Dynamic Range, US patent 5801773, Sept. 1998.
2. S.K. Nayar, M. Watanabe, and M. Noguchi, "Real-Time Focus Range Sensor," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, 1996, pp. 1186- 1198.
3. M. McGuire et al., "Defocus Matting," *ACM Trans. Graphics*, vol. 24, 2005, pp. 567–576.
4. M. Aggarwal and N. Ahuja, "Split Aperture Imaging for High Dynamic Range," *Int'l J. Computer Vision*, vol. 58, no. 1, 2004, pp. 7-17.
5. R.P. Harvey, Optical Beam Splitter and Electronic High-Speed Camera Incorporating Such a Beam Splitter, US patent US5734507, 1998.
6. K.-H. Tan, H. Hua, and N. Ahuja, "Multiview Mirror Pyramid Cameras," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 7, 2004, pp. 941-946.
7. B. Wilburn et al, "High-Speed Video Using a Dense Camera Array," *Proc. Computer Vision and Pattern Recognition*, IEEE CS Press, 2004.
8. T. Mitsunaga and S. Nayar, "High Dynamic Range Imaging: Spatially Varying Pixel Exposures," *Proc. Computer Vision and Pattern Recognition*, IEEE CS Press, vol. 1, 2000, pp. 472-479.
9. S. Nayar and S. Narasimhan, "Assorted Pixels: Multisampled Imaging with Structural Models," *Proc. European Conf. Computer Vision*, 2002, p. 636.

We use the configuration from Figure 4 designed by the optimizer to capture hybrid high-speed visible/infrared video. A hot-mirror directs infrared down the right subtree and visible light down the left subtree. Each subtree has four cameras with temporal phase offsets, so the entire system yields 120 fps video with four spectral samples. Figure 14 shows frames from a sequence in which a person catches a tumbling infrared remote control and then transmits it at the camera. Because the configuration captures four spectral samples at 120 fps, the high-frequency infrared pattern transmitted by the remote is accurately recorded, as is the fast tumbling motion at the sequence's beginning.



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

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 microbalance the true response curves of components and are frequently more efficient.

For future work, we're investigating multispectral HDR for surveillance, alternative multifocus approaches for matting, and multispectral high-speed video 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. We believe that genetic algorithms are likely to produce better output than our optimizer because they're naturally suited to a tree combination.

The optical elements form a filter system that terminates at digital sensors. 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. P.E. Debevec and J. Malik, "Recovering High Dynamic Range Radiance Maps from Photographs," *Proc. Siggraph*, ACM Press, 1997, pp. 369-378.
2. T. Mitsunaga and S. Nayar, "High Dynamic Range Imaging: Spatially Varying Pixel Exposures," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition* (CVPR), vol. 1, IEEE CS Press, 2000, pp. 472–479.
3. M. Aggarwal and N. Ahuja, "Split Aperture Imaging for High Dynamic Range," *Int'l J. Computer Vision*, vol. 58, no. 1, 2004, pp. 7-17.
4. S.K. Nayar, M. Watanabe, and M. Noguchi, "Real-Time Focus Range Sensor," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 12, 1996, pp. 1186- 1198.
5. S. Chaudhuri and A. Rajagopalan, *Depth from Defocus: A Real Aperture Imaging Approach*, Springer Verlag, 1998.
6. Y. Xiong and S. Shafer, "Depth from Focusing and Defocusing," *Proc. Computer Vision and Pattern Recognition,* IEEE CS Press, 1993, pp. 68-73.
7. M. Subbarao, "Parallel Depth Recovery by Changing Camera Parameters." *Proc. 6th Int'l Conf. Computer Vision* (ICCV), IEEE CS Press, 1998, pp. 149-155.
8. M. McGuire et al., "Defocus Matting," *ACM Trans. Graphics*, vol. 24, 2005, pp. 567-576.
9. P. Debevec et al., "A Lighting Reproduction Approach to Live-Action Compositing," *ACM Trans. Graphics*, vol. 21. no. 3, 2002, pp. 547-556.

Morgan McGuire *is an assistant professor of computer science at Williams College. His research interests include multisensor imaging, video as an input device, real-time rendering, real-time physics, and the engineering of rule systems for puzzles and games. McGuire received his BS and MEng in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT) and his MS and PhD in computer science from Brown University. Contact him at morgan@cs.williams.edu.*

Wojciech Matusik *is a research scientist at Mitsubishi Electric Research Laboratories (MERL). His primary interests are computer graphics, data-driven modeling, computational photography, and new display technologies. Matusik received the following degrees in electrical engineering and computer science: a BS in from the University of California at Berkeley, an MS from MIT, and a PhD from MIT. Contact him at matusik@merl.com.*

Hanspeter Pfister *is a senior research scientist at MERL. His research is at the intersection of computer graphics, visualization and computer vision. Pfister has a PhD in computer science from the State University of New York at Stony Brook. He is chair of the IEEE Visualization and Graphics Technical Committee (VGTC). Contact him at pfister@merl.com.*

Billy Chen *is a software development engineer for the Virtual Earth team at Microsoft. His research interests include image-based modeling, computational photography, and projector-camera systems. Chen received a BS in electrical engineering and computer science from the University of California, Berkeley, and an MS and PhD in computer science from Stanford University. Contact him at bill.chen@microsoft.com.*

John F. Hughes *is an associate professor of computer science at Brown University. His interests include geometric modeling, expressive rendering, sketch-based interfaces, the interface between computer vision and computer graphics, and machine learning. Hughes received a BA in mathematics from Princeton University and an MA and PhD in mathematics from the University of California, Berkeley. Contact him at jfh@cs.brown.edu.*

Shree K. Nayar *is the T.C. Chang Chaired Professor in the Department of Computer Science at Columbia University. He's also the codirector of the Columbia Vision and Graphics Center and heads the Computer Vision Laboratory. His research is focused on three broad areas—the creation of novel vision sensors, the design of physics-based models for vision, and the development of algorithms for scene interpretation. Nayar received his PhD in electrical and computer engineering from the Robotics Institute at Carnegie Mellon University. Contact him at nayar@cs.columbia.edu.*