RTcams: A New Perspective on Non-Photorealistic Rendering from Photographs

Peter M. Hall, John P. Collomosse, Yi-Zhe Song, Peiyi Shen, and Chuan Li

***Abstract*— We introduce a simple but versatile camera model that we call the Rational Tensor Camera (RTcam). RTcams are well principled mathematically and provably subsume several important contemporary camera models in both Computer Graphics and Vision; their generality is one contribution. They can be used alone, or compounded to produce more complicated visual effects. In this paper we apply RTcams to generate synthetic artwork with novel perspective effects from real photographs. Existing Non-Photorealistic Rendering from Photographs (NPRP) is constrained to the projection inherent in the source photograph, which is most often linear. RTcams lift this restriction and so contribute to NPRP via multi-perspective projection. This paper describes RTcams, compares them to contemporary alternatives, and discusses how to control them in practice. Illustrative examples are provided throughout.**

***Index Terms*— Generalized Cameras, Non-Photorealistic rendering, Projective systems.**

I. Introduction

THE ideal pin-hole camera generates images in linear perspective. Having been well studied and extensively used in both Science and Art, its properties are well known. Recently, non-linear projective cameras have been researched in both Computer Graphics and Computer Vision, driven by a variety of motivations. Our motivation came from noticing that linear perspective is surprisingly rare in Art: children, draughtsmen, architects and engineers regularly draw using non-linear projections, either by accident or design. The Oriental and African schools, amongst other non-Western schools, make little if any use of linear perspective. Historically, the pre-Renaissance and post-impressionist Western Art also use non-linear perspective. We set out to introduce a wide variety of perspective effects into real photographs. The output is processed into paintings using standard Non-Photorealistic Rendering from photographs (NPRP). We wanted a simple method, able to emulate not only real cameras but also perspective types typical to Art; hence RTcams.

Our work was in part motivated by our own observations that Art often uses non-linear projection and also by Art commentator John Willats. He differentiates the *projective system* that is concerned with perspective in artwork, from the *denotational system*[1] that is concerned with the way marks are made [1]. Willats shows both systems play an important role in categorizing a piece of artwork. The lesson for NPRP is that to convincingly emulate a style of art, it is not sufficient to consider either system alone; both must be considered.

NPR covers a wide range of research topics including multi-perspective rendering, artistic rendering, cartoon rendering, and so on. The literature on NPRP, specifically, focuses overwhelmingly on denotational systems. Sometimes by computing the form or placement of brush strokes, other times painting media is modeled. Despite significant progress in these areas, NPRP algorithms remain restricted to painting over projections from real cameras, and the largest and most important class of real cameras can be approximated to first-order by a pin-hole camera. The significance of this is that NPRP has been restricted in the gamut of projective styles it exhibits. We conclude that NPRP faces what might be called a "projective barrier".

Our contribution is to lift the projective barrier by considering the projective system, rather than the denotational system. We introduce Rational Tensor Cameras, or RTcams for short (the pun on arty-cameras is intentional). RTcams provide a versatile family of non-linear cameras. RTcams can model lens aberrations in real cameras and can even be calibrated to them, accounting for barrel and pin-cushion aberrations. Equally, they enable photographs to be processed into projective forms that no physical camera can ever capture but which are used by human artists. The images output from RTcams retain a photographic texture and so can be used as fancy photos, or they can be painted over with any NPRP algorithm. In this way we can build a system that takes both the (multi)perspective and denotational systems into account.

Non-linear cameras remove the requirement that rays of light must pass through a single point focus. In fact non-linear cameras may contain any number of real-valued focal points, including zero or infinitely many. They are the subject of a considerable volume of contemporary study and are used in many different ways in both Graphics and Vision. Contemporary models include X-slit [2] cameras, push-broom cameras [21], strip cameras [3]–[6], General Linear Cameras [20] and Rational Function cameras (RFcams) [7]. The next Section provides a fuller review. RTcams are tensor based cameras that emulate (and can be calibrated to) many other non-linear cameras including all those just mentioned. RTcams can combine different photographs from different views into one, or can be used to define a light field. The important features of RTcams are that they:

- Provide a single, simple model of a camera that not only contains several important, contemporary models of generalized cameras as special cases, but have their

Manuscript received August 2006.

All authors are with the Media Technology Research Centre, Department of Computer Science, University of Bath, BA2 7AY, UK. Email {pmh|jpc|yzs20|cssps|cl249}@cs.bath.ac.uk. Ph +44(1225)386811. Fax +44(1225)383493

[1]Willats's division is more general than the special case given here, but the projection/mark-making division is valid and directly applicable to NPRP.

unique properties (perspective effects) too.

- Introduce a wide variety of projection effects. Many of these are impossible to achieve with any real camera but may be seen in the work by artists of all skills, ages, and many schools.
- May be combined easily, to produce larger *compound* RTcams. Compound RTcam may themselves be combined, *ad infinitum*. This allows users to construct complex optical devices that include light beam splitting and merging.

Examples of the kind of effects RTcams can produce can be seen in Figure 1. We now give a brief review of related literature.

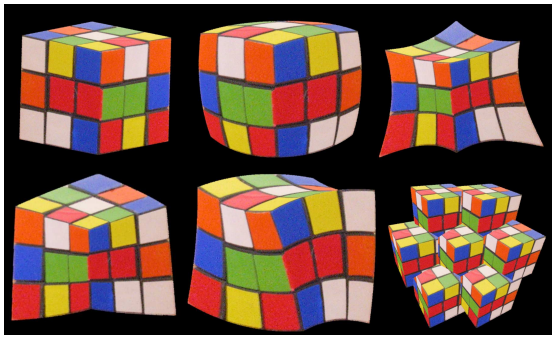


Fig. 1. Example applications of RTcams to a Rubik's cube. Top row: The identity RTcam (left) leaves the image unchanged, here perspective projection from a real camera. A barrel (middle) and pin-cushion (right) distortion. Bottom row: inverse perspective (left), a depth-dependent twist (middle), and a compound "fly-eye" lens effect (right) created with a compound RTcam. Close inspection of the latter reveals that cubes are not stacked to form a wall, but that cubes intersect to form a regular hexagon, and that the same cube is viewed 7 times, each from a slightly different angle.

A. Background

This section provides a brief review of multi-perspective rendering. X-slit cameras have been used to mosaic new views without recovering 3D geometry and without camera calibration [2]; the foci of an X-slit camera lie on a pair of lines making an X-shape. Rademacher and Bishop [3] construct multi-view panoramas of a kind suitable for cel animation, using a *strip camera*. Strip cameras have appeared quite often in the Graphics literature, with the aim of mosaicing photographs. Roman *et al*. [4], [5] provide a semi-interactive system that uses a linear camera to combine photographs into panoramas of street scenes, as do Agrawala *et al.* [6]. The latter authors contribute by reducing the degree of user interaction to identifying the dominant plane. In all cases linear camera models are used. RTcams can emulate strip cameras using ordinary photographs, a non-linear camera and minimal user interaction; see Subsection V-A for an example.

Rational Function cameras (RFcams) [7] are used to measure and correct radial distortions, such as pin-cushion or barrel deformations, that obtain from real cameras. They operate in the window plane of the camera by mapping homogeneous pixel coordinates using rational quadratic functions, and can be calibrated to real cameras. Methods based on tensors have been proposed for calibrating generic non-linear cameras [8]. Such calibration methods mean that RTcams too can be calibrated, see the Appendix for details. Calibration not only allows RTcams to emulate real cameras but means RTcams can be calibrated to emulate other non-linear cameras. Calibration allows a user interface to be built making them easier to use. Such an interface simply allows the user to draw a warped rectangular grid; the warping away from true rectilinearity is used to calibrate the RTcam.

NPRP is kin to 3D-model based NPR, where research into non-linear projection is currently active. Polygons can be rendered non-linearly on standard hardware via an adapted form of scan-conversion [9], [10]. But ray-tracing is the predominant method by which to non-linearly render models. Loffelmann and Groller [11] use an "extended camera" in which points on the model each have an associated ray. Levene [12] provides a set of heuristics, such as allowing image surfaces (windows) — which are planar in the pin-hole camera — to be polynomial surfaces. Glassner advocates the use of "putty lenses" [13], again within a ray-tracer. Agrawala *et al*. [14] discuss the issue of how to depth-order a set of models when each is seen from a unique point of view. Specifically, when the same point is seen from many views, which depth should be used when computing hidden surfaces? They propose a "master camera" as a means of resolving ambiguities. Karan Singh and his co-workers have contributed much to this area of the literature [15], [16]. Coleman and Singh show that linear interpolation is a solution to the problem of viewing a single model from more than one viewpoint. They too use the idea of a master camera [16]. More recently, they provide widgets to assist user interaction [17].

The Generalized Linear Camera (GLC) of Yu and McMillan [20] specifies a light field; each ray in the field is a linear combination of three basis rays. GLCs can emulate X-slit cameras as well as others such as push-broom cameras [21], but cannot model pin-cushion or barrel distortion. GLCs have many applications, including light-field rendering and 3D-model based NPR, their versatility and simplicity make them interesting. Similarly, Mei *et al.* [22], define a field of light by specifying a variety of geometric terms related to ordinary pin-hole cameras (focal length, center of interest, etc) on a pair of planes. Their *occlusion camera* is more complex than GLCs, but is specifically designed to capture occlusion boundaries of 3D models.

John Willats worked with Fredo Durand [18] using 3D models to demonstrate that his descriptions of artistic style can be connected to rules used to generate pictures. Halle [19] also uses multi-viewpoint rendering, not to produce a combination of several views in one, but rather to efficiently produce many similar views.

II. RTcams: with a broad brush

We begin our more detailed discussion with a brief discussion of the overall rendering pipeline we advocate. This can be seen in Figure 2. Although RTcams form only a part of this pipeline, and although they contribute to multi-perspective rendering only, it is important to provide such a pipeline. This is because our motivation is the emulation of perspective

as used by artists in different times, at different places, of different ages, and so on. Our pipeline is consistent with Willats's [1] division of style into projective and denotational systems, its simplicity is seen as an advantage.

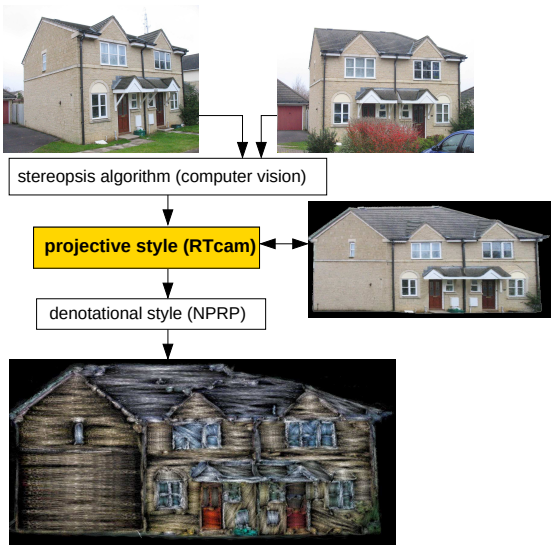


Fig. 2. We address the projective barrier facing NPRP by using standard stereopsis to create a point cloud model (not necessarily a full reconstruction, see text for details). This is acted on by RTcams so the *projective style* of the artwork can be selected; this is the contribution of this paper. RTcams produce photographic quality output which can be painted into a *denotational style* using state-of-the-art NPRP algorithms.

We change the projective system using stereo photography. This is preferred over a single image because it allows much greater freedom in warping — relative depth turns out to be useful. Stereo photography allows us to build a partial model of the scene, which can then be re-photographed — in our case with an RTcam. Stereopsis has been applied previously in NPRP, but to better highlight edges [23], [24] which is an issue of the denotational system, rather than to address the projective system as we do.

The Computer Vision literature has a vast literature devoted to reconstruction from a stereo pair. The majority of it recovers points located in three dimensions. Fitting lines, planes, and other geometric objects takes extra effort and implies extra assumptions too. RTcams are designed to work with points — the simplest possible geometric primitive. Stereo relies on corresponding pixels in the two given images. Mis-matched pixels lead to erroneous 3D locations. The importance of these outliers to our application depends on many factors, which makes a full discussion out of place here. The most serious issue here is that outlying points project to arbitrary positions in the final image. We have found that hand-based segmentation greatly assists the process of corresponding points, so that outliers are not an issue for us in practice.

In this paper, the reader may assume an object comprises a cloud of points. Importantly, this cloud need not be a full reconstruction. Computer vision recognizes three classes of reconstruction: (i) Perspective reconstruction is the broadest class in which connectivity between points is preserved and straight lines remain straight. A cube will typically appear as a highly distended shape in which each face, though flat, is squashed more at one edge, appearing exactly as under linear perspective. (ii) Affine reconstructions preserve length ratios — a cube may be reconstructed with faces which are parallelograms. (iii) Euclidean reconstructions preserve angles too; a cube is reconstructed correctly, up to a scale ambiguity; see Faugeras and Luong [25].

RTcams are neutral with respect to the class of reconstruction. Perspective, affine, and Euclidean reconstructed point clouds are all treated equally. The class of reconstruction needed depends on the details of the application. For example Figures 10 and 13 both require Euclidean reconstructions of a house, but Figure 12 and Figure 14 need only perspective reconstructions. Section V has details.

The remainder of this paper introduces background mathematics to understand RTcams (Section III). We go on to discusses practical issues in Section IV — motivating the need for *control spaces*, and *compound RTcams*. Section V provides specific examples that illustrate RTcams can emulate a wide gamut of artwork. The appendix discusses how RTcams relate to alternative non-linear cameras, proving their generality, and shows how RTcams can be calibrated from matched points.

III. Rational Tensor Cameras: Some Mathematical Background

In this section we define and discuss the mathematical essentials of RTcams. We use the notational convention that $\Re^n$ refers to an n dimensional vector space, and P^n is the corresponding projective space. We generate P^n from $\Re^n$ in the usual way: by defining a basis vector that is orthogonal to $\Re^n$ so that points $\mathbf{x} = [x_1, \ldots, x_n] \in \Re^n$ require a "homogeneous coordinate" to be appended: $\mathbf{x} \rightarrow x_n\lambda[\mathbf{x}\ 1]$ for $\lambda \neq 0$. The additional vector is, of course, the "homogeneous direction" and the value of a point's homogeneous coordinate is its "homogeneous depth".

As a note, all of the photographic examples in this paper use RTcams defined in P^3, which means the objects are three-dimensional. The visualizations though, that is Figures 4 and 5, use RTcams defined in P^2. This is so homogeneous depth can be visualized as a third spatial dimension. The mathematics makes no assumption about the dimensionality of the space.

We begin by defining an RTcam. An RTcam is a projection $\mathcal{Q} : P^n \mapsto P^n$. We write $\mathbf{y}(\mathbf{x}) = \mathcal{Q}[\mathbf{x}]$. The RTcam maps a vector $\mathbf{x} \in P^n$, to another vector $\mathbf{y} \in P^n$ using a ratio of tensor operations on the input vector $\mathbf{x}$. In this paper we use third-order tensors, which are $(n \times n \times n)$ cubes of numbers. The ith element in the output vector is computed as

$$y_i(\mathbf{x}) = \frac{\mathbf{x}\mathbf{Q}_i\mathbf{x}^T}{\mathbf{x}\mathbf{Q}_n\mathbf{x}^T} \tag{1}$$

in which $\mathbf{Q}_i$ is a $(n \times n)$ matrix, which can be thought of as a plane of the tensor, see Figure 3. Each $\mathbf{Q}_i$ has exactly the same effect as its transpose, so that we can assume $\mathbf{Q}_i$ is symmetric without loss of generality.

This definition of an RTcam is a direct analog of the standard linear camera, which is specified using a single $(n \times n)$ matrix, $\mathbf{C}$. If $\mathbf{C}_i$ is the ith row of $\mathbf{C}$, then the ith element in the projection is given by $y_i = (\mathbf{x}\mathbf{C}_i^T)/(\mathbf{x}\mathbf{C}_n^T)$. The linear camera is therefore the ratio of linear functions. As

we will see below, RTcams are ratios of quadratic functions. As with linear cameras, it is convenient to think of projection as acting in two steps — a geometric mapping followed by a perspective projection:

$$z_i(\mathbf{x}) = \mathbf{x}\mathbf{Q}_i\mathbf{x}^T \quad (2)$$

$$y_i(\mathbf{z}) = \frac{1}{z_n}\mathbf{z} \quad (3)$$

We say the *object* $\mathbf{x}$ is projected to the *image* $\mathbf{y}$. We call $\mathbf{z}$ the *homogeneous image* so as to tell it apart from $\mathbf{y}$. The orthogonal projection of the image onto a plane generates points that can be visualized on a computer, we use $[y_1, y_2]$, and call this point the *visible image*. The action of a specific RTcam in P^2 is shown in Figure 4. It shows how the RTcam warps points in an object from a plane onto a quadric surface, which is then projected. The result is an image which appears non-linearly distorted.

Because RTcams operate in projective space they are invariant to scale: $\mathbf{x} \equiv \lambda\mathbf{x}$. We can take advantage of this by fixing the homogeneous depth of input points, $x_n = 1$. The allows us to partition a matrix in the tensor, $\mathbf{Q}$ into a scalar part, a linear part, and a quadratic part, which together comprise a Taylor expansion, specifically

$$\begin{aligned} z(\mathbf{x}) &= [\mathbf{x}, 1]\mathbf{Q}[\mathbf{x}, 1]^T \\ &= [\mathbf{x}, 1]\begin{bmatrix} \mathbf{H} & \mathbf{L}^T \\ \mathbf{K} & s \end{bmatrix}[\mathbf{x}, 1]^T \\ &= \mathbf{x}\mathbf{H}\mathbf{x}^T + (\mathbf{K}\mathbf{x}^T + \mathbf{x}\mathbf{L}^T) + s \\ &= s + \mathbf{x}\mathbf{J}^T + \mathbf{x}\mathbf{H}\mathbf{x}^T \end{aligned} \quad (4)$$

with $\mathbf{J} = \mathbf{K} + \mathbf{L}$. We identify the non-linear part of the camera $\mathbf{H}$ as the Hessian, the linear part $\mathbf{J}$ as the Jacobian, and the scalar s as the constant at $z(\mathbf{0})$; see Figure 3. Hence RTcams are the ratio of quadratic functions, as claimed above.

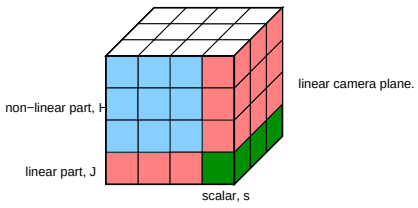


Fig. 3. The tensor for an RTcam is a cube of numbers. Each plane is a matrix mapping an object into a single element of the image. The mapping comprises a scalar constant, a linear part and a non-linear part that can be expanded in a Taylor series. The "side plane" of the cube is the matrix for a linear camera.

Furthermore, we can think of a linear camera as modeling a real camera to a first order approximation, and an RTcam as adding a second-order correction. Exactly this correction is used by by Claus and Fitzgibbon [7] to calibrate and correct for lens aberrations in real cameras. Those authors confine themselves to P^2 whereas typically we use RTcams in P^3. We also include tools designed specifically for NPRP uses: control spaces in Subsection IV-B and compound cameras in Section IV-C. It is worth noting that some lens aberrations do in fact depend on real depth, Seidel aberrations for example [26], and that RTcams do allow for depth dependent effects and so can approximate these.

The decomposition of the transform into independent parts — scalar, linear, quadratic — means users can consider each of these terms independently, which makes control of RTcams much easier. For example, we see that setting $\mathbf{H} = 0$ gives the special case of a linear camera; each column of the linear camera's matrix $\mathbf{C}$ is just a Jacobian. The fact that RTcam matrices can be replaced with a symmetric version reduces the number of parameters needed to specify an RTcam from n^3 to $n^2(n+1)/2$, so providing further assistance. The next subsection provides mathematical elements of RTcams, needed to helps one understand their relation to other camera models. Practical issues resume in Section IV.

A. Further Analysis

The definition of RTcams given above is sufficient for general purpose use. This subsection provide a more detailed mathematical analysis so that its operation can be more fully understood, and so that RTcams can be compared with other non-linear cameras.

RTcams are homogeneous degree 2, because for any scalar λ we have $z_i(\lambda\mathbf{x}) = \lambda^2 z_i(\mathbf{x})$. It follows that for any $\lambda \neq 0$, $y_i(\lambda\mathbf{x}) = y_i(\mathbf{x})$. Therefore the image is invariant to a uniform scaling of the object.

Linear cameras contain a single focus and a plane of points that they cannot project because $\mathbf{x}\mathbf{C}_n^T = 0$. Points in this singular plane are "invisible" to the camera, in the sense that the camera cannot make an image of them. RTcams generalize from the linear camera case. The quadratic equation

$$\mathbf{x}\mathbf{Q}_n\mathbf{x}^T = 0 \quad (5)$$

denotes a quadric surface which is invisible to an RTcam. We call this surface the *surface at infinity*, because it is made up of points $\mathbf{x} \in P^n$ at infinity that map to $\mathbf{z}$ such that $z_n = 0$. The absolute origin $\mathbf{0}$ is a point on the surface at infinity. The non-degenerate solutions are more interesting, not least because they may have complex elements. Promoting real-valued vectors to have complex valued components is called the *complexification* of projection space, Faugeras and Luong [25]. Briefly, we can see that $\mathbf{x}\mathbf{Q}_n\mathbf{x}^T = 0$ may have complex roots by using eigenvalue decomposition of $\mathbf{Q}_n$ to get $\mathbf{x}\mathbf{R}\mathbf{S}_n\mathbf{R}^{-1}\mathbf{x}^T = 0$. We can now map the object vector into the eigenbasis to get $\mathbf{v}\mathbf{S}_n\mathbf{v} = \sum_i v_i^2 s_{ii} = 0$. Since the s_{ii} are positive, the v_i must be complex in general.

The surface at infinity is important because it characterizes not only a particular camera, but also the class of camera. For example, no RTcam can represent a camera with a surface at infinity that cannot be described by a quadric. This can be used to build a taxonomy of cameras based on sub-class relations between surface families. For example, the bilinear surfaces are a sub-class of the quadric surfaces because all bilinear surfaces are representable using a quadric, but not vice-versa. It follows that because the surface at infinity of GLCs are bilinear surfaces, they can be considered as a special case of RTcams. The Appendix discusses the relationship between RTcams and other cameras in greater detail.

Linear cameras map straight lines to straight lines. RTcams map straight lines to quadratic curves. Suppose $\mathbf{x} = \mathbf{p} + s\mathbf{u}$

is an arbitrary parametric straight line in $\Re^n$. The line is such that $x_n = 1$, so is in a subspace of P^n. Under an RTcam, this line maps to the quadratic curve in P^n given by

$$\begin{aligned}\mathbf{z}(s) &= \mathbf{x}\mathbf{Q}_i\mathbf{x}^T + \mathbf{x}(\mathbf{Q}_i + \mathbf{Q}_i^T)\mathbf{u}^T s + \mathbf{u}(\mathbf{Q}_i + \mathbf{Q}_i^T)\mathbf{u}^T \frac{s^2}{2} \\ &= \mathbf{z}(0) + \frac{d\mathbf{z}(0)}{ds}s + \frac{1}{2}\frac{d^2\mathbf{z}(0)}{ds^2}s^2 \end{aligned} \quad (6)$$

This is easy to show by differentiating Equation 2. All derivatives of $\mathbf{z}$ of third and higher order are zero. Figure 4 shows how straight lines map to quadratic curves.

By analogy with Faugeras and Luong [25], we define the *vanishing points* of a line in P^n as the images of its points at infinity. The vanishing points on a line defined as in Equation 6 are easy to find: we simply solve the quadratic equation $z_n(s) = 0$ to obtain s values. Although these are complex in general they can then be used to obtain vanishing points.

We define a *ray* of light to be the locus of object points $\mathbf{x}$ in $\Re^n$ that project to the same image point $\mathbf{y}$. Note that because RTcams map straight lines to curves in projective space, a ray can appear to bend in $\Re^n$. (A ray in P^n is just a straight line emanating from the origin.) We require the differential structure of the RTcam; the matrix of partial derivatives is

$$\frac{\partial y_i}{\partial x_j} = \frac{1}{z_n^2}\left(z_n\frac{\partial z_i}{\partial x_j} - z_i\frac{\partial z_n}{\partial x_j}\right) \quad (7)$$

in which $\mathbf{z}$ is the homogeneous image. The total change in $\mathbf{y}$ due to an infinitesimal change in $\mathbf{x}$ is therefore

$$dy_i = \frac{\partial y_i}{\partial x_j}dx_j \quad (8)$$

We use Einstein's convention for tensors — repeated indices denotes summation. The matrix of partials depends on object location; call it $\mathbf{P}(\mathbf{x}) = \partial y_i/\partial x_j$. It is rank degenerate because the column vector $\partial y_n/\partial x_j = 0$ for all j. Consequently there is at least one direction vector such that $d\mathbf{y} = \mathbf{P}d\mathbf{x} = 0$, that is which leaves the image $\mathbf{y}$ stationary. We call this tangential direction an *instantaneous ray*; integrating over these recovers the ray. Given a point $\mathbf{x}$, the direction $\lambda\mathbf{x}$ is an instantaneous ray, because it leaves $\mathbf{y}$ stationary. If the rank of $\mathbf{P}$ is $n-1$, then the instantaneous ray through a point has a unique direction. If the rank is lower than this, then there may be many instantaneous rays through the same point. All vanishing points have $\mathbf{P}$ of rank 0. The row vector $\partial y_i/\partial x_n$ is the instantaneous ray direction, projected into image space. The radial lines in Figure 4 are rays.

IV. Arty cameras: RTcams applied to Non-Photorealistic Rendering

Recall that in this paper RTcams use input vectors produced via stereopsis. We remind ourselves that RTcams affect perspective and do not care whether the points come from a perspective, affine, or Euclidean reconstruction. We create artistic looking images by painting over a photograph with non-linear projection, created by an RTcam, as explained in Section II. Having outlined the mathematics of RTcams already, it remains for us to explain how to use them in practice. This section therefore gives several example RTcams,

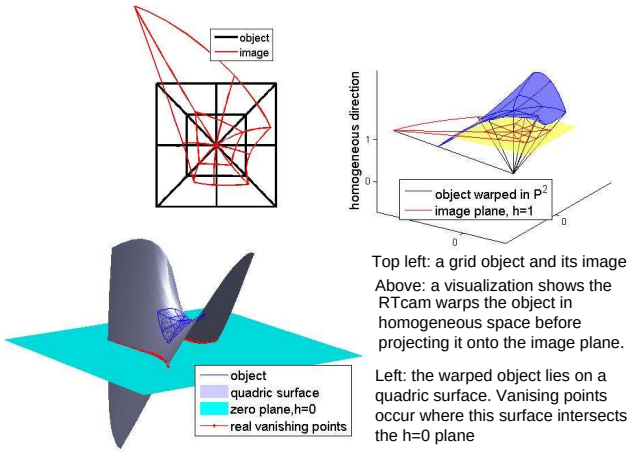


Fig. 4. An RTcam in P^2 (a 2D camera in homogeneous space)

introduces the concept of *control space* as a means to control effects, explains how to combine RTcams to generate a *compound RTcam*, describes a user interface, and briefly discusses issues germane to rendering such as anti-aliasing and hole-filling.

A. Specific RTcam examples

At first glance, the action of an RTcam may seem difficult to specify, but just as one learns to specify matrices so one can learn to specify tensors. We provide some examples in this section. But first note that by using the Taylor expansion we can easily recreate a linear camera. Given a matrix $\mathbf{C}$ we distribute its ith column, scaled by $1/2$, into the final column and bottom row of the ith RTcam matrix $\mathbf{Q}_i$. The effect is to put $\mathbf{C}$ into a plane of the tensor that crosses each of the $\mathbf{Q}_i$, see Figure 3. The remaining terms in a particular $\mathbf{Q}_i$, the "upper-left" corner, account for all non-linearities.

An RTcam that produces barrel distortion (see Figure 1) under the control of a parameter s is:

$$\mathbf{Q}_1 = \begin{bmatrix} 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 \end{bmatrix} \mathbf{Q}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 \\ 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 \end{bmatrix}$$

$$\mathbf{Q}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 0 \end{bmatrix} \mathbf{Q}_4 = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The first three matrices are the identity transforms, in that given $[x_1, x_2, x_3, 1]$ they produce $[x_1, x_2, x_3]$. The final matrix, which determines homogeneous depth, is responsible for the non-linear effect. It gives $z_n = s(x_1^2+x_2^2+x_3^2)+1$; if $s = 0$ we obtain the identity RTcam. If s is large, a large barrel effect is produced. If s is negative, we get pin-cushion distortion. Both can be seen in Figure 1. Clearly, the surface at infinity, $z_n = 0$, of this RTcam will have complex roots, when $s > 0$, because then we require $x_1^2 + x_2^2 + x_3^2 = -1/s$.

Inverse perspective causes objects to appear to dilate, rather than diminish, with distance. This effect is possible with an RTcam, but of course no real camera can produce this. Inverse perspective can be found in the Byzantine art, the work of Matisse, the Cubists, in children's artwork and elsewhere. We can obtain a dilation with real depth, x_3 say, by considering the mapping element x_i in the object vector to $x_i + x_i \alpha x_3$, for some α, and in which x_3 is the depth of the object in real space. In this case we set only the first two RTcam matrices:

$$\mathbf{Q}_1 = \begin{bmatrix} 0 & 0 & \frac{\alpha}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ \frac{\alpha}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \end{bmatrix} \mathbf{Q}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\alpha}{2} & \frac{1}{2} \\ 0 & \frac{\alpha}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

The remaining matrices are set up to produce $z_3 = x_3$ and $z_4 = 1$. We note that setting $\alpha = 0$ yields an orthogonal projection. The result of applying inverse perspective is demonstrated in Figure 1.

A depth dependent twist is our final example, produced by means of a cross product. The first two matrices are

$$\mathbf{Q}_1 = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{\alpha}{2} & 0 \\ 0 & \frac{-\alpha}{2} & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \end{bmatrix} \mathbf{Q}_2 = \begin{bmatrix} 0 & 0 & \frac{\alpha}{2} & 0 \\ 0 & 0 & 0 & \frac{1}{2} \\ \frac{\alpha}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

which on expansion give $z_1 = x_1 - (\alpha x_3)x_2$ and $z_2 = x_2 + (\alpha x_3)x_1$ respectively. If we set $\sin(\theta)/\cos(\theta) = \alpha x_3$, then we see $z_1 = x_1 \cos(\theta) - x_2 \sin(\theta)$, and $z_2 = x_2 \cos(\theta) + x_1 \sin(\theta)$, which is a depth dependent twist, up to a scale; the output vector length is scaled by $(1 + \alpha^2 x_3^2)$. To eliminate this we would have to redefine RTcams to take the square root of the denominator, so that input and output vector lengths remain constant. One may opt to modify the RTcam definition by setting $z_i = (\mathbf{xQx})^{\beta_i}$, but in this paper we retain the simplicity of the original definition. To complete the twisting RTcam, we set $\mathbf{Q}_3$ and $\mathbf{Q}_4$ to return x_3 and x_4 respectively. Figure 1 shows a depth dependent twist, but applied in all directions at once.

The final example in Figure 1 will not be explained here, because creating a compound-eye effect requires several RTcams to be combined. Exactly how to do this is explained later, but first we will consider controlling RTcam effects a little more closely using various spaces.

B. Control spaces

The visual effect of an RTcam on an object depends on where in space that object is, as well as the values in the tensor. Picking values is in principle no more difficult than specifying matrix transforms. The effect of the spatial location of objects is managed using *control spaces*. Figure 5 shows an example for a two dimensional grid experiencing barrel distortion at various locations in the plane. It is clear that the output depends critically on the location of the input. As we explain below, defining a *canonical space* as a particular control space, a user can specify the canonical action of an RTcam, and store it in a library for later use, making RTcams easier to use.

Conventional 3D computer graphics recognizes "world space" , "image space" and "object space", amongst others. Doing so allows users to exercise control over models and cameras. Effects such a scaling and rotation, for example, are typically best done in an object's own frame of reference, that is in object space, whereas hidden surface problems may be most easily solved in image space. Conversion from one space to another is effected by linear transforms, often arranged into a hierarchy.

RTcams are transforms, and it is convenient to recognize different spaces, and their respective advantages and disadvantages. The general idea conforms exactly to standard practice: a map transforms the object into some convenient space, the desired transform is applied to create the image, and finally an inverse map is used to carry the image back into the original space.

Suppose the matrix $\mathbf{M}$ is a linear mapping that carries an object $\mathbf{x}$ into the control space to give $\mathbf{x}' = \mathbf{xM}$, both being homogeneous points. This new point is subject to the transform we wish to apply. We neglect to divide by the homogeneous distance but nonetheless write $\mathbf{z}' = \mathcal{Q}[\mathbf{x}']$. Neglecting to divide is in line with common practice when mapping between spaces. The transformed vector $\mathbf{z}'$ is subject to the inverse mapping to obtain the homogeneous image of the object: $\mathbf{z} = \mathbf{z}'\mathbf{M}^{-1}$. All vectors belong to P^n.

If the RTcam mapping were linear, then all of these transforms can be collected into a single transform; we need only multiply the matrices to get the single matrix $\mathbf{R} = \mathbf{MQM}^{-1}$ — this "concatenation" is a well known advantage of using matrices. It turns out we can do the same when $\mathcal{Q}[.]$ is an RTcam, although the process is a little more complicated. It can be shown that by setting

$$\mathbf{R}_j = (\mathbf{M}^T \mathbf{Q}_i \mathbf{M}) \mathbf{M}_{ji}^{-1} \tag{9}$$

the RTcam R performs the mapping into the control space, applies the RTcam Q, and performs the inverse mapping out of the control space: that is, R applied Q in the control space. Again, repeated indices imply summation, so each $\mathbf{R}_j$ is a weighted sum of the $(\mathbf{M}^T \mathbf{Q}_i \mathbf{M})$.

Returning to the discussion of named spaces, we retain the concept of a *world space* as the default space in which homogeneous points reside. This space is useful for tasks such as rotation and translation but is not well suited for applying scaling and non-linear effects to objects, for example. The analogy with object space is harder to maintain, because objects may not be Euclidean reconstructed. However, we anticipate that users will want to apply RTcams in the reference frame of the camera, not the object. This makes sense because it is the visual effect of the camera that is interesting.

When constructing a canonical space in the frame of the camera we first make an *image space*. This is directly analogous to the space of the image space defined by pixels — points map to pixels under parallel projection (Figure 6). We then map image space to canonical space using a scale and shift, using an RTcam to do so. This RTcam is defined by fitting a bounding cone around an object's points. The apex of the cone is the absolute origin. The sides of the cone lie along the line of sight. Two parallel planes, each of constant homogeneous depth, cap the cone at either end. The whole cone is skewed and scaled into a cylinder that

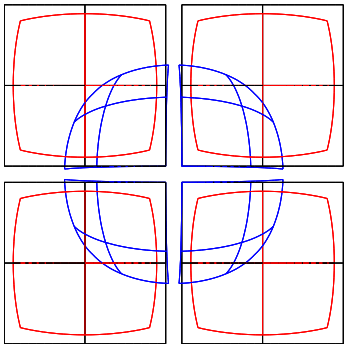


Fig. 5. An example showing the effect of an RTcam on an object depends on where the object is. An object is made of four grids (black). The blue image is the effect of the barrel camera without transforming the object into canonical space. The red image is the effect when canonical space is used on each grid.

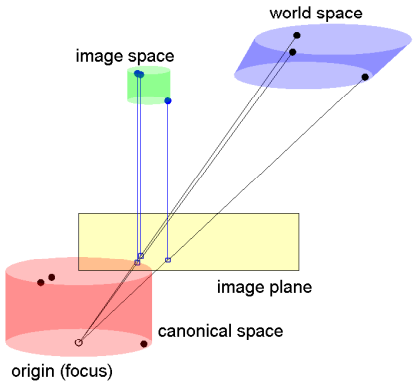


Fig. 6. Control spaces: Points in world space (blue) are mapped into points in image space (green), which maps convergent rays from a real camera into parallel rays. Points in image space are mapped canonical space (red), which makes many effects easier to control because the points are enclosed by a known geometry. The yellow plane is the image plane, which remains in a constant location.

projects orthogonally to the same set of points. This RTcam can be expressed with an invertible, linear operation. The newly formed cylinder is mapped to *canonical space*. Another invertible linear transform shifts and scales the cylinder so that the center of the near plane is at the origin, and the cylinder has unit height and radius. Now that the object lies in a well defined region the effects become much more predictable, placing RTcams under user control.

C. Compound RTcams

So far we have considered RTcams as tensor transforms. Much can be done with these, as shown in the examples of Figure 1 and some of the extended examples in Section V. Even greater power can be achieved by combining the *atomic* RTcams studied so far to create *compound RTcam*. So far as we know, compound cameras are a unique contribution of this paper, even though others combine (linear) cameras. Coleman and Singh [16] show how weighted cameras can be used to combine many views of many linear cameras into one. We too use a linear combination, but of non-linear cameras. Moreover, we combine cameras in serial and in parallel, to create a complicated structure, represented by directed acyclic graph (DAG). This vastly widens the gamut of possible effects.

Compound RTcams are DAGs. Each node contains an RTcam which processes points in an object. The arcs of the DAG govern the flow of the data (set of points) between nodes, see Figure 7. A simple DAG has two nodes, A and B say, arranged in series so that A is a parent node and B a child node. A set of points is input to this compound RTcam via node A, which produces a new set of points that is input to the RTcam in node B. The output of B is the output of the compound RTcam. It is easy to create a series of RTcams of any length. The example in Subsection V-B depends, in part, on RTcam in series.

A different case arises when node A has two children, B_1 and B_2, say. In this case the output of A is input to both children, which are in parallel. Extension to N parallel children is easy. Equally, a node, C say, can have M parents in parallel. We call A a "splitter" and C a "merger". The problem is in the merging node, which must combine multiple images, one from each parent, each showing the object from different points of view. Here a *view* means the image of an object under any RTcam, so scaling, rotating, barrel-distortions are all *views*. There is no unique solution to the problem of combining multiple views, so we discuss and demonstrate two methods.

The first is straight-forward — the point sets output by each child are concatenated into a single set. This produces multiple copies of the same object. The problem of depth ordering can be solved using a z-buffer. This is how the compound-eye example in Figure 1 was made. The second method is to merge views by linearly interpolating the points in them so as to create a set of points. This is a more complicated method, but one which is rewarding because it enables a single image to show the same object from quite different views, simultaneously, with each point appearing at most once in the final image.

1) Merging views by interpolation: When merging point sets from nodes in parallel it is important our implementation of RTcams does not change point ordering. This seeming triviality make it easy to identify corresponding points in images: the ith point in data set j corresponds to the ith point in data set k and so on (the j and k index the parent nodes; as B_1 and B_2 are parents to C, for example). Suppose $\mathbf{x}_{ij}$ is the ith point in the jth data set to be merged. Linear interpolation gives

$$\mathbf{y}_i = \sum_{j=1}^{N} w_{ij}\mathbf{x}_{ij} \tag{10}$$

as the ith output point, where w_{ij} is a weight associated with the ith point of the jth input set; we require $\sum_j w_{ij} = 1$. Setting $w_{ij} = w_j$, a constant for the jth view usually gives results of little interest, the general case is far more interesting and is discussed next.

The aim is to specify a set of weights w_{ij}, one weight for every point in every data set. In Figure 7 there are two views to merge, so each weight vector has two elements. This could

be an overwhelming task were it not for our user interface that allows users to “paint the weights”. The idea is very simple: users paint the object being images by the compound RTcam. Our method is very similar to one described by Coleman and Singh [16], whose RYAN system combines views in a hierarchy of linear cameras. We explain the system here for completeness.

An object is painted in three dimensions with the intention of color coding regions. For example, in Figure 7 the user has outlined regions in red and in green which are easy to fill with solid color. Not all points in the object are painted, but those which are painted are viewed from a single point of view, specifically from the view of the camera associated with the color code. In the example of Figure 7 these are the orthogonally projected front (red) and side (green) views. Fixed views can be easily arranged by setting the vectors w_{ij}. In the example, if the front view corresponds to the $j = 1$ view we need only set $w_{i1} = [1, 0]$ for those points painted red, while $w_{i2} = [0, 1]$ for those painted green. This scheme extends in a natural way to more than two views.

Weights for points which have no fixed view (that is, have not been painted by the user) are generated as follows. First, all points in the object are projected to a reference view. The painted points make up a set of colored regions on this view. Next each unpainted point i is considered; the closest distance to each colored region is found so that a vector of distances d_{ij} at is created. The reciprocal of distance, when normalized, gives the weight we seek: $w_{ij} = (1/d_{ij})/\sum_j(1/d_{ij})$. We can choose any point of view for the reference projection, and the particular projection chosen will affect the weights on the unpainted point — but we have found that almost any projection gives acceptable results so this has not been a major issue for us. What is more significant is that the unpainted regions of an object are deformed in the final output image in such a way as to fit exactly between the fixed (painted regions). In the example of Figure 7 the roof is stretched to fit snugly between the front and side views. Willats [1] claims this is how children might draw a house, a claim which motivated this example.

2) An algorithm for compound RTcams: Our algorithm for using a compound RTcam is now given. First place a single object at each root of the DAG, a root node has no parents. Then perform a breadth-first traversal of the DAG by processing the point set through each node. In general, each node will merge its input sets from each parent, apply the node’s RTcam, and split the output to each of its children. Each node has a flag which, if set, maps the points into canonical space before RTcam application and inverts the mapping afterwards. A second flag in the node indicates the way multiple images are combined: concatenation or merging. The point sets from the leaves (having no children) are concatenated by default to create a single image.

Compound RTcams take a single object (picture) as input and give a single object as output. Therefore, compound RTcams can be treated as atomic RTcams. Now atomic RTcams are defined in projective space, so we can interpret object points as rays, if we wish. This is helpful in thinking about compound RTcams because a set of points can equally well

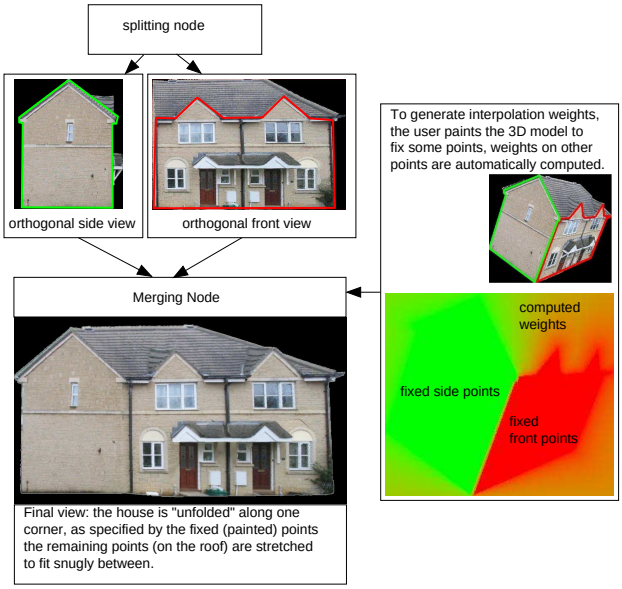


Fig. 7. A compound RTcam is used to photograph a house from two different views, simultaneously. A single input object is seen from two views, which are combined under user control to obtain an output image.

be thought of as a set of rays, or a “beam”. By analogy, then, we can think of a compound RTcam as a generalized optical device in which RTcams are lenses, and nodes act as both beam splitters and beam mergers.

V. RTcam: Creating and Rendering Examples

Earlier we mentioned the possibility of segmenting an image into pieces and applying an RTcam to each, but chose instead to consider the case where points arise from stereopsis. Neither automated segmentation nor stereopsis give satisfactory results without user supervision, and user interaction is to set weights used in compound RTcams. In addition, users will probably wish to build a library of RTcams. We have built a user interface specifically designed for use with RTcams, a snapshot of this interface is shown in Figure 8. This user interface allows users to easily segment images into parts that are meaningful to them, they can provide assistance to improve the results of stereopsis, create and assign RTcams including setting weights for combining different views, and control the denotational style in which the output is rendered.

Importantly, our interface allows a user to assemble a library of RTcams that operate in canonical control space. New RTcams can be defined simply by drawing down the desired effect on a rectangular grid. The grid provides a set of points that can be used to automatically set RTcam parameters — an easy way to define RTcams. This automatic process relies on a calibration process (see Appendix) that is can also be used to calibrate an RTcam to a given real camera. A full description of our user interface is beyond the scope of this paper. We continue by considering rendering issues.

RTcams produce a set of points to be rendered. The task at hand now is to render the points into an image made of pixels. We face three problems. First, a consequence of the non-linear mapping is that the points may not be uniformly

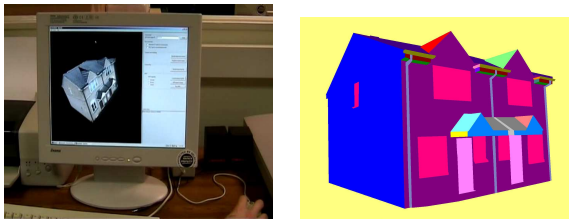


Fig. 8. A user interacts with the RTcam system, positioning local RTcams (left) and assigning them to regions of interest in the source photographs using "magic scissors" (right).

spread, leaving "gaps" in the rendering that must be filled by interpolation. These "gaps" can be filled automatically. A "gap" is identified as being an uncolored pixel close to at least 4 other colored pixels, and within the convex hull of those pixels. The color of the gap pixel is determined by interpolation. We anti-alias using Gaussian point-splatting. The second problem is that the rendering may contain "holes" caused by parts of the object that were occluded in each of the original photographs used for stereopsis, and "holes" may appear between warped objects. Both kinds of holes must be filled in using texture-filling.

Fig. 9. An example of raw output from an RTcam. Object parts that were obscured in the source photos appear as holes, which are filled-in using by growing surrounding texture into them. Anti-aliasing takes the non-uniform distribution of points into account, as explained in the text.

The final step of the rendering process is to optionally apply existing NPRP algorithms to the image. This allows us to use a denotational system more sympathetic to the projective system. For example, Figure 7 shows how we can "flatten" the front and side view of a house into a single image. We have mentioned already that Willats [1] claims this "flattening" is used by children, who then fit the roof wherever it may land. This is exactly what our example does. Yet the output in Figure 7 is not convincing as a child's drawing, because the denotational style is photographic. Automatic over-painting in crayon [27] yields Figure 10, in which denotational and projective styles match to produce a house rendering that is convincingly child like.

We now illustrate the versatility of RTcams by providing examples emulating a range of different artistic styles, each one of which requires non-linear projection to be convincing.

A. Combining views: a 'Northern school' projection

Artist David Hockney [28] suggests, amidst some controversy, that artists of the Northern schools may have used optical devices. The artist Vermeer is widely reputed to have

Fig. 10. *My House*: A crayon rendering generated over a child-like projection.

used a camera obscura. By drawing, Hockney demonstrates that the same object can have more than one vanishing point, which he explains by the artist moving the optical device as they worked. Hockney's observations have influenced his recent work, made by pasting together photographs taken from different views similar to an example in Figure 11.

Our aim here is to merge photographs to create a single image with a subtly varying view point, typical of the Northern school. An atomic RTcam suffices, because we use it to interpolate between two points of view. Suppose we have two linear cameras, one with a projection matrix $\mathbf{A}$, the other has projection matrix $\mathbf{B}$. In Section III we showed an RTcam can be expanded as a Taylor series, so that a linear camera can be emulated by filling in the linear parts appropriately. If $\mathbf{A}_i$ is the ith column of the camera, we put

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{0} \\ \mathbf{A}_i^T \end{bmatrix}$$

then force symmetry, if we wish $\mathbf{Q}_i \leftarrow \frac{1}{2}(\mathbf{Q}_i + \mathbf{Q}_i)$ To interpolate between views we interpolate over the vertical axis of the image, so use

$$\mathbf{Q}_i = \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_i^T - \mathbf{A}_i^T \\ \mathbf{0} \\ \mathbf{A}_i^T \end{bmatrix}$$

again forcing symmetry. In any case, an expansion of the above, using the input point $\mathbf{x} = [x_1, x_2, x_3, 1]$ gives $y_i = \mathbf{x}((\mathbf{B}_i - \mathbf{A}_i)x_2 + \mathbf{A}_i)$ Now each output value is the interpolation of two cameras, with x_2 being the control variable. This variable comes from the reconstruction of the objects, we need only a perspective reconstruction. In this regard we echo the work of Zomet *et al.* [2] who use cross-slit camera to achieve similar results (but cross-slit camera cannot reproduce the many other effects of RTcams). Because we know x_2 scans the vertical dimension of the image we can normalize it to vary between 0 and 1. Each $\mathbf{Q}_i$ is right multiplied by a matrix $\mathbf{S}$ that scales and shifts x_2 from its domain, $[1, N]$ say to index into N scan-lines, into the range $[0, 1]$, so we can replace x_2 in the above with $u = (x_2 - 1)/(N - 1)$.

The mathematics can be interpreted in two equivalent ways: (1) the point's position in camera $\mathbf{A}$ and camera $\mathbf{B}$ is interpolated, or (2) a single camera is moved from the position of camera $\mathbf{A}$ to the position is camera $\mathbf{B}$. The latter interpretation emulates a strip-camera. The equivalent strip-camera is initially placed coincident with camera A, and the

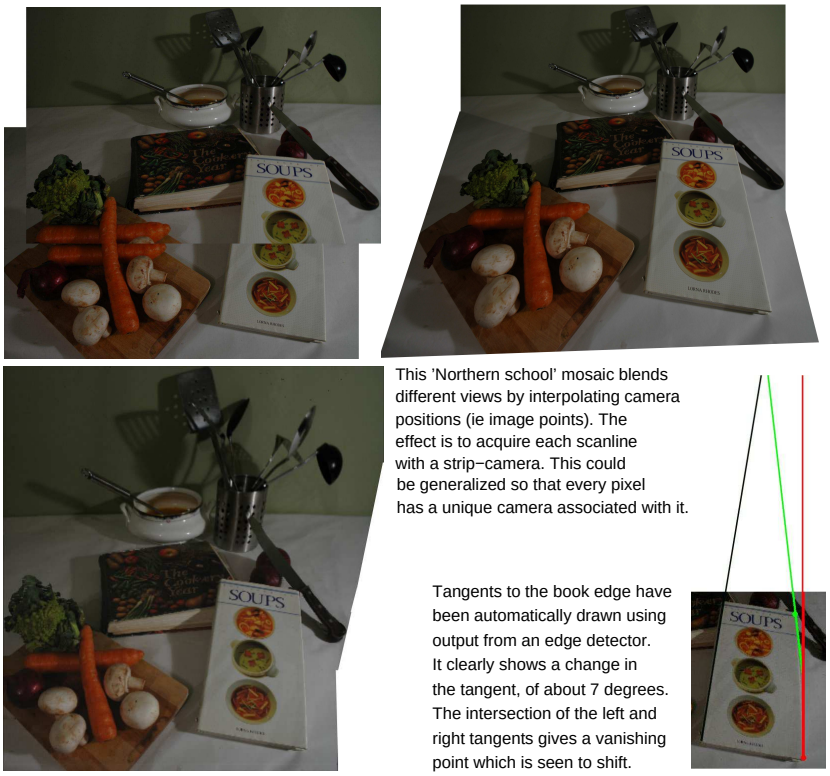


Fig. 11. Three ways to mosaic two images: top-left is the Hockney-esque approach of pasting one photo over the other, top-right is a mosaic constructed by determining the best-fit homography between the images and warping one to match. Bottom-left is the result of our method. Our method can be thought of as emulating a strip-camera. It produces results more in line with the projective system of the Northern school, as we desired: straight lines, such as the edge of a book, are bent almost imperceptibly.

bottom-row of pixels is copied into the target. The strip-camera is then moved just a little toward B, and the second to bottom scan-line is copied, and so on until the camera B is reached, where the top scan-line is copied.

There is a subtlety to be explained. There was not enough information in the original images to extract 3D points of the vegetables in the near-plan view, and the utensils in the other photograph. This does not prevent the interpolation from being applied to them, but means that the objects are treated as flat planes at infinity — much as the photographs in conventional mosaicing are considered as planes at infinity. This simplification is acceptable under a projective reconstruction and necessary if these complex objects are to change perspective with the rest of the image and so avoid the need for filling gaps between objects.

The result of merging is seen in Figure 11, which also includes both a Hockney-esque and standard panoramic mosaicing of merged images for comparison. The objects in the scene are too close to the cameras for standard panoramic mosaicing algorithms to work well (the plane-at-infinity assumption breaks down). Our 'Northern school' merge is subtle, and we think is of higher quality than the standard panoramic mosaic. The near imperceptible change in perspective view point moving up the page cause straight lines to bend. To illustrate this bending we draw tangents to the side of a book automatically, using edge detection and line fitting. The change in tangent is a measure of the change in vanishing point. Once holes around the edge of the image have been filled, and the

Fig. 12. *"Still Life"* in the projective and denotational style of the Northern school.

result painted, we obtain an emulation that is consistent with the projective and denotational styles of the Northern school, see Figure 12.

Before leaving this example, we remark that the ability to create panoramas such as this raises the interesting possibility

of using RTcams to create cel panoramas from ordinary cameras for input, rather than the real strip-cameras used by Rademacher and Bishop [3]. Zomet *et al.* [2] mosaic in way similar to our own, using X-slit cameras, yet RTcams are more general then X-slit camera.

B. Serial RTcams: The haunted house

The Northern school example used a single RTcam. Here we use a compound RTcam with nodes in serial, that is one after the other in a DAG. It is motivated by noticing Artists' skill in creating different moods using both projective and denotational systems. For example, art work in comic books may exaggerate or even invert perspectives for dramatic effect, straight lines may be drawn as curves. This tradition is seen too in films and television; villains in the 1960's version of Batman hang out in dens filmed at peculiar angles and the "Psycho" house was manufactured about $\frac{3}{4}$ real size, and filmed from below. This exaggerated perspective and helped build a tense atmosphere around the Bates motel.

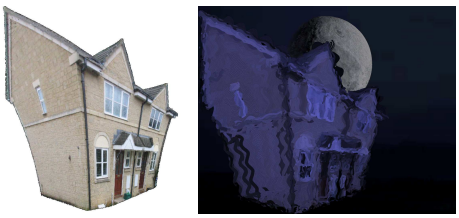


Fig. 13. Left: Photographic output of a compound RTcam. Right: *"Haunted house"* in comic-book style.

We created Figure 13 using the same house photographs as in Subsection V (the child's drawing). Both examples require Euclidean reconstruction. The compound RTcam used here comprises two atomic RTcams in series. The first applies inverse perspective, the second a pin-cushion distortion in three dimensions and views the house from low down, giving a threatening appearance.

To complete the ambiance, the house has been composited onto a spooky landscape and heavily stylized using a oil painterly rendering algorithm [27] where stroke tone has been automatically modulated to enhance the creepiness of the image. Figure 13 shows both the photographic and painterly stylized output. This example also shows that the same model, in this case a house, can be rendered in many different ways, tuned to a particular application by the combination of both projective and denotational styles — both are needed to create convincing artistic effects.

C. RTcams in parallel: a Byzantine mosaic

Mosaicing with tiles is a traditional form of picture making. Our example copies a projective system seen in a mosaic of a vase from the Byzantine school. The bottom of the vase is viewed "front on", so that it looks like a straight line. The mouth of the vase is viewed at a steep angle, using a depth dependent skew. A compound DAG merges these views by interpolation. Finally, we used an existing algorithm [29] to synthesize the denotational style of a mosaic. Figure 14 shows both the photographic and mosaiced results.

Fig. 14. Left: Photographic output of vase seen under a "Byzantine" RTcam. Right: *"Vase from Byzantium"*.

The handles of the vase were cut out manually prior to any RTcam application, and re-composited afterwards. This hints at the value of a *compositional system*, which refers to the relative location, orientation and size of objects in pictures for esthetic effect, rather than the technical problem of compositing. The compositional system can be used to make a scene appear perpetually stable, or unstable, for example. Eastern Art uses a compositional framework that differs from that of Western Art: more distant objects are placed higher up the page. The next example continues the compositional theme.

D. Compositing parts: Expressionist projection

Our final example partitions a scene into objects and applies a different RTcam to each, to create an image in projective and denotational styles that emulate expressionism. The expressionists broke many rules, including those of linear perspective. Our example is derived from Matisse's *Harmony in Red*. Matisse used orthogonal projection to emphasize the table, tilting its objects to show them in canonical views. We have developed an algorithm that automatically and easily chooses canonical points of view [30].

Following Matisse, we created three objects from a scene, one for a table-top, one for a cup, one for a bowl of fruit. A different atomic RTcam was applied to each, to depict it in orthogonal projection, and seen from a canonical or near-canonical point of view. The output images composited were to create the final picture seen in Figure 15. As usual, we painted over the photographic output in an appropriate painterly style [27].

We found re-compositing the transformed parts a little awkward. This is because their shape and size has to be carefully fitted into the surrounding elements of the scene. RTcams having nothing to say about compositional issues, and we are not aware of any study that does. We conclude that "esthetic composition" is a major open issue in NPRP.

VI. Discussion and Concluding Remarks

RTcams are a non-linear camera model that contributes to multi-perspective rendering. Multi-perspective rendering is an

Fig. 15. Top: Photographic output in the projective style of Matisse. Bottom: *"Harmony in style"*.

important component of NPR; this is the first time we are aware of that multi-perspective has been used specifically in NPRP. RTcams address the "projective barrier" facing NPRP, and highlight the division between projective and denotational systems.

Atomic RTcams are second-order rational tensors, that subsume several other camera models available, including: GLCs, strip-cameras, X-slit cameras, and the rational function cameras used to correct for radial aberrations in real cameras (see Appendix). Each of these cameras is able to reproduce part of the RTcam repertoire, but none can reproduce all of it, and RTcams are capable of unique effects both in atomic and compound form. Camera models that RTcams cannot emulate include the occlusion camera [22], which requires division by a square root term. Generally, RTcams can model any camera whose surface at infinity is contained in the set of all quadric surfaces.

When modeling real cameras we can think of RTcams as a second order correction to a first order approximation (the linear camera). Higher order corrective terms require higher-order tensors. Fourth-order tensors give cubic functions, and a cubic surface at infinity. Another way to generalize RTcams is to pass both the numerator and demoninator through some function, so as to raise each term to some power, or to take its logarithm or exponential. These generalizations would allow RTcams to emulate a much wider class of cameras, including occlusion cameras. In this paper, we elected to follow neither of these possibilities. Instead we generalized by introducing compound RTcams, allowing for highly non-linear aggregation of simpler cameras. Compound RTcams are able to split and merge beams (sets of points), and so are analogous to optical devices. So far as we know the idea of building compound optical devices is new.

We have used RTcams in several examples: showing how to merge two photographs using both a single RTcam and a compound RTcam. The examples demonstrate the importance of both denotational and projective systems in defining the esthetics of the final artwork. A particular lesson from the "vase" and "Matisse" examples is the importance of what we call the *compositional system*, which appears to have been over-looked by Willats's [1]. We have encountered compositional issues before, when we emulated Cubism [31] and Futurism [32] — experience which adds conviction to our proposition. We therefore propose it makes sense to modify Willat's taxonomy to include this additional system.

In summary, RTcams contribute to multi-perspective rendering by providing a simple camera model with a strong mathematical base; they unify many important, contemporary non-linear camera models. Although conceived to address the "projective barrier" facing NPRP, RTcams can model real camera aberrations. Because RTcams can be compounded into complicated non-linear optical devices the gamut of reachable projective styles becomes very wide. By bringing the projective system under user control, RTcams enable the re-creation of perspective effects typically seen in real artwork, many of which cannot be reproduced with a real camera.

References

[1] John Willats, *Art and representation : new principles in the analysis of pictures*, Princeton University Press, 1997.

[2] A. Zomet, D. Feldman, S. Peleg, and D. Weinshall, "Mosaicing new views: The crossed-slits projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 6, pp. 741–754, 2003.

[3] P. Rademacher and G. Bishop, "Multiple-center-of-projection images," in *Proc. 25$^{\text{th}}$ ACM SIGGRAPH*, New York, NY, USA, 1998, pp. 199–206, ACM Press.

[4] A. Roman, G. Garg, and M. Levoy, "Interactive design of multi-perspective images for visualizing urban landscapes," in *IEEE Visualization*, 2004.

[5] A. Roman and H.P.A. Lensch, "Automatic multiperspective images," in *Eurographics Symposium on Rendering*, 2006.

[6] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski, "Photographing long scenes with multi-viewpoint panoramas," in *ACM SIGGRAPH*, 2006, pp. 853–861.

[7] D. Claus and A.W. Fitzgibbon, "A rational function lens distortion model for general cameras," in *Computer Vision and Pattern Recognition*, 2005, pp. 213–219.

[8] S. Ramalingam, P. Sturm, and S.K. Lodha, "Towards complete generic camera calibration," *IEEE CVPR*, vol. 1, pp. 1093–1098, 2005.

[9] V. Popescu, J. Eyles, A. Lastra, J. Steinhurst, N. England, and L. Nyland, "The warpengine: An architecture for the post-polygonal age," in *ACMSIGGRAPH*, 2000, pp. 433–442.

[10] X. Hou, L-Y Wei, H-Y Shum, and B. Guo, "Real-time multi-perspective rendering on graphics hardware," in *Eurographics Symposium on Rendering*, 2006.

[11] H. Loffelmann and E. Groller, "Ray tracing with extended cameras," *The Journal of Visualization and Computer Animation*, vol. 7, no. 4, pp. 211–227, 1996.

[12] J. Levene, "A framework for non-realistic projections," M.S. Thesis, MIT M.Eng., 1998.

[13] A.S. Glassner, "Cubism and cameras: Free-form optics for computer graphics," Tech. Rep., Microsoft Research, 2000.

[14] M. Agrawala, D.Zorin, and T Munzner, "Artistic multiprojection rendering," in *Eurographics Workshop on Rendering Techniques*. 2000, pp. 125–136, Springer-Verlag, London.

[15] K. Singh, "A fresh perspective," in *Graphics Interface*, 2002, pp. 17–24.

[16] P. Coleman and K. Singh, "RYAN: Rendering your animation non-linearly projected," in *Non-photorealistic Rendering and Animation*, 2004, pp. 129–138.

[17] P. Coleman, K. Singh, L. Barrett, C. Grimm, and N. Sudarsanam, "3d screen space widgets for nonlinear projection," in *ACM GRAPHITE*, 2005, pp. 221–228.

[18] J. Willats and F. Durand, "Defining pictorial style: Lessons from linguistics and computer graphics," *Axiomathes2005*, vol. 15, no. 2, 2005.
[19] M. Halle, "Multiple viewpoint rendering," in *ACM SIGGRAPH*, 1998, pp. 243–254, ACM Press.
[20] J. Yu and L. McMillan, "A framework for multipersective renderings," in *Eurographics Symposium on Rendering*, 2004.
[21] R. Gupta and R.I. Hartley, "Linear pushbroom cameras," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, pp. 963–975, September 1997.
[22] C. Mei, V. Popescu, and Elisha Sacks, "The occlusion camera," in *Computer Graphics Forum (Eurographics)*, 2005, vol. 24.
[23] R. Raskar, K-H. Tan, R. Feris, J. Yu, and M. Turk, "Depth edge detection and stylized rendering using multi-flash imaging," *Proc.* 31^{st} *ACM SIGGRAPH*, vol. 23, no. 3, pp. 679–688, 2004.
[24] E. Stavrakis and M.Gelautz, "Stereoscopic painting with varying levels of detail," in *Proc. SPIE*, 2005, vol. 5664, pp. 450–459.
[25] O. Faugeras, Q-T Luong, and T. Papadopoulo, *The geometry of multiple images*, MIT Press, 2001.
[26] P. Brakhage, G. Notni, and R. Kowarschik, "Image aberrations in optical three-dimensional measurement systems with fringe projection," *Applied Optics*, vol. 43, no. 16, pp. 3217–3223, June 2004.
[27] M. Shugrina, M. Betke, and J. P. Collomosse, "Empathic painting: Interactive stylization using observed emotional state," in *4th Intl. Symposium on Non-photorealistic Animation and Rendering*, 2006, pp. 87–96.
[28] D. Hockney, *Secret knowledge : rediscovering the lost techniques of the old masters*, Thames and Hudson, 2001.
[29] G. Di Blasi and G. Gallo, "Artificial mosaics," *The Visual Computer*, vol. 21, no. 6, pp. 373–383, June 2005.
[30] P. Hall and M.J. Owen, "Simple canonical views," in *British Machine Vision Conference*, 2005, pp. 839–848.
[31] J. Collomosse and P. Hall, "Cubist style rendering from photographs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 9, no. 4, pp. 443–453, 2003.
[32] J. Collomosse and P. Hall, "Genetic paint: A search for salient paintings," in *EvoMUSART*, 2005, p. (to appear).

APPENDIX

RTCAMS AND SOME CONTEMPORARY NON-LINEAR ALTERNATIVES

Here we compare RTcams to several non-linear alternatives, showing that RTcams are more general versions of each. We begin with the rational function cameras introduced by Claus and Fitzgibbon [7] to compensate for radial aberrations in real cameras. They too use the ratio of quadratic functions, but use an RTcam defined in P^2. We use RTcams in P^3 and are therefore more general. Similar remarks apply to the X-slit cameras of Zomet*et al.* [2], who use tensor based projection. Again, our tensors are more general and we conclude RTcams are more general than X-slits.

Comparison with General Linear Cameras (GLCs) [20] is more difficult (and so takes more space). This is because GLCs define the direction of a set of rays, whereas RTcams operate using points. GLCs operate in P^2 by defining a ray using three basis rays. Each basis ray is specified by a pair of points, each of three elements because the camera acts in P^2. The points $\mathbf{r}_i$ and $\mathbf{s}_i$ define i the basis ray. The $\mathbf{r}_i$ form a triangle in the plane of zero homogeneous depth (so $r_{i3} = 0$ for all i), the $\mathbf{s}_i$ all have unit homogeneous depth ($s_{i3} = 1$ for all i). Three numbers, an input point, specify any particular point $\mathbf{z}(\alpha, \beta, \gamma)$, γ being the distance along the ray.

$$\begin{aligned}\mathbf{z}(\alpha,\beta,\gamma) = \alpha\mathbf{r}_1 + \beta\mathbf{r}_2 + (1-\alpha-\beta)\mathbf{r}_3 + \gamma(\alpha(\mathbf{s}_1-\mathbf{r}_1)\\ +\beta(\mathbf{s}_2-\mathbf{r}_2) + (1-\alpha-\beta)(\mathbf{s}_3-\mathbf{r}_3))\end{aligned}$$

By setting $\mathbf{u}_i = \mathbf{s}_i - \mathbf{r}_i$, which is a basis ray direction, and writing each vector as a row in a matrix, we can express the above as

$$\mathbf{z}(\alpha,\beta,\gamma) \quad = \quad [\alpha\ \beta\ 1]\left(\begin{bmatrix}\mathbf{p}_1\\ \mathbf{p}_2\\ \mathbf{p}_3\end{bmatrix} + \gamma\begin{bmatrix}\mathbf{v}_1\\ \mathbf{v}_2\\ \mathbf{v}_3\end{bmatrix}\right)$$

in which we define the matrices

$$\mathbf{P} = \begin{bmatrix}\mathbf{r}_1-\mathbf{r}_3\\ \mathbf{r}_2-\mathbf{r}_3\\ \mathbf{r}_3\end{bmatrix}\ \mathbf{V} = \begin{bmatrix}\mathbf{u}_1-\mathbf{u}_3\\ \mathbf{u}_2-\mathbf{u}_3\\ \mathbf{u}_3\end{bmatrix}$$

The ith element of the output point is now

$$z_i(\alpha,\beta,\gamma) \quad = \quad \alpha p_{i1} + \beta p_{i2} + p_{i3} + \gamma(\alpha v_{i1} + \beta v_{i2} + v_{i3})$$

The equivalent RTcam is specified in P^3. For each *matrix* $i = 1, 2, 3$

$$\mathbf{Q}_i = \begin{bmatrix}\mathbf{0} & 0\\ \mathbf{0} & 0\\ \mathbf{p}_i & 0\\ \mathbf{v}_i & 0\end{bmatrix}\quad \mathbf{Q}_4 = \begin{bmatrix}\mathbf{0} & 0\\ \mathbf{0} & 0\\ \mathbf{e}_3 & 0\\ \mathbf{0} & 0\end{bmatrix} \tag{11}$$

in which $\mathbf{e}_3$ is the unit row vector $[0, 0, 1]$. The above definition is asymmetric; symmetry may be forced upon all $\mathbf{Q}_i$ by halving the sum of each matrix and its transpose, but this is not required. Using the asymmetric version we see that

$$z_i(\alpha,\beta,\gamma) \quad = \quad \alpha p_{i1} + \beta p_{i2} + p_{i3} + \gamma(\alpha v_{i1} + \beta v_{i2} + v_{i3})$$

which is identical to the GLC. This shows any GLC can be represented by an equivalent RTcam. Furthermore, the RTcam has three additional degrees of freedom in the "top-left corner" of each matrix with index $i \in [1, 2, 3]$, preventing GLCs from emulating every RTcam. Therefore RTcams are more general than GLCs.

GLCs define rays on a bilinear surface, which is reflected in the structure of the RTcam matrices above. The surface at infinity for GLCs is therefore bilinear, and because these are a sub-class of quadric surfaces, GLCs are special case RTcams. This is an alternative proof the RTcams subsume GLCs.

Specifying a ray, as GLCs do, is necessary for Computer Graphics applications such as ray-tracing. But ray-tracing is not an an efficient rendering strategy for the point clouds make our models. It is much more efficient to project the points along the ray that passes through it. The problem facing GLCs in this context is determining the ray direction though an arbitrary point in space, the GLC must be "inverted". Such an inversion is not found in the GLC literature, but we give it here for completeness. We omit any proof, supplying only the main results which may be verified by the reader.

The ith coordinate of the ray vector through a point is given by a ratio of quadratic equations, that is by an RTcam with matrix planes given by the tensor equations $\mathbf{Q}_i = v_{ji}\mathbf{D}_j$ and $\mathbf{Q}_4 = \mathbf{D}_3$ in which the $\mathbf{D}_i$ are (4×4) matrices, defined below. Given the RTcam as specified above, the ray at $\mathbf{x}$ is $w_i = \mathbf{x}^T\mathbf{Q}_i\mathbf{x}/\mathbf{x}^T\mathbf{Q}_4\mathbf{x}$ The first three coordinates of the ray $\mathbf{w}$ (i.e. an orthogonal projection) is the ray passing through the point, but represented within a GLC. This ray carries the point $\mathbf{x}$ onto the image point given by $\mathbf{y} = \mathbf{x} + x_3\mathbf{w}$ which is

identically the image point in the GLC. The $\mathbf{D}_i$ are matrices defined by

$$\mathbf{D}_1 = \frac{1}{2}\begin{bmatrix} 0 & 0 & v_{22} & p_{22} \\ 0 & 0 & -v_{21} & -p_{21} \\ v_{22} & -v_{21} & b_1 & a_1 \\ p_{22} & -p_{21} & a_1 & c_1 \end{bmatrix}$$

$$\mathbf{D}_2 = \frac{1}{2}\begin{bmatrix} 0 & 0 & -v_{12} & -p_{12} \\ 0 & 0 & v_{11} & p_{11} \\ -v_{12} & v_{11} & b_2 & a_2 \\ -p_{12} & p_{11} & a_2 & c_2 \end{bmatrix}$$

$$\mathbf{D}_3 = \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & b_3 & a_3 \\ 0 & 0 & a_3 & c_3 \end{bmatrix}$$

In which we define vectors $\mathbf{a}$, $\mathbf{b}$, and $\mathbf{c}$

$$\begin{aligned} \mathbf{b} &= 2(\mathbf{v}_1 \otimes \mathbf{v}_2) \\ \mathbf{c} &= 2(\mathbf{p}_1 \otimes \mathbf{p}_2) \\ \mathbf{a} &= (\mathbf{p}_1 \otimes \mathbf{u}^2) - (\mathbf{p}_2 \otimes \mathbf{u}^1) \end{aligned}$$

We use the notation $\mathbf{p}_i$ to refer to the ith column of $\mathbf{P}$, and $\mathbf{v}^i$ to refer to the i row of $\mathbf{V}$; other terms in the above follow by analogy. The ability to compute rays, and hence image points is sufficient to specify an RTcam. This is because given a sufficient number of object/image point matches we can calibrate an RTcam, as explained next.

CALIBRATING RTCAMS

We wish to determine an RTcam given a set of object points, $\mathbf{x}$, and a set of corresponding image points $\mathbf{y}$. We begin by re-writing $z_i = \mathbf{x}\mathbf{Q}_i\mathbf{x}^T$ as $z_i = (\mathbf{x}^T\mathbf{x}) \odot \mathbf{Q}_i$ in which $\odot$ multiplies corresponding matrix elements and takes the sum — it is an inner product. Setting $\mathbf{U} = \mathbf{x}^T\mathbf{x}$, and recalling $\mathbf{Q}_i$ has a symmetric equivalent $\mathbf{P}_i$ allows us to write the above as the familiar inner product of vectors, each with $n^2(n+1)/2$ elements: we write $z_i = \mathbf{u}\mathbf{p}_i^T$. The relation with components of the image point $\mathbf{y}$ can now be expressed as

$$\mathbf{u}\mathbf{p}_i^T - y_i\mathbf{u}\mathbf{p}_n^T = 0$$

It is the RTcam elements, the $\mathbf{p}_i$ we seek, which we therefore factor out to yield a homogeneous set of equations of the form

$$\begin{bmatrix} \mathbf{u}_j & \mathbf{0} & -y_{1j}\mathbf{u}_j \\ \mathbf{0} & \mathbf{u}_j & -y_{2j}\mathbf{u}_j \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} = \mathbf{0}$$

in P^2 and an analogous form for P^3; the j indexes the jth pair of matched points. Provided, in P^n, there are more than $n^2(n+1)/(2(n-1))$ matches, the singular valued decomposition of the design matrix of $\mathbf{u}_j$ and $\mathbf{y}_j$ yields a null left-singular vector which is readily converted to the solution $\mathbf{P}$, up to a scale factor that makes no difference to the camera's operation. In this way the RTcam can be calibrated, with applications such as "inverse" GLC emulation by an RTcam, or as means by which users can specify an RTcam by drawing its effect.

AUTHOR BIOGRAPHIES

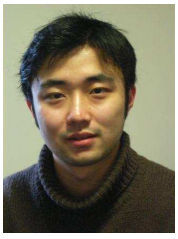
This work was supported by EPSRC grant EP/D064155/1.

Peter Hall is a senior lecturer (associate professor) in Computer Science at the Univ. of Bath where is belongs to the Media Technology Research Centre (MTRC). He took a PhD in 1993, from Sheffield Univ., studying under Dr. Alan Watt. He regularly publishes in both Computer Vision and Computer Graphics literature. He is chair of the Vision, Video and Graphics community, and an executive member of the British Machine Vision Association.

John Collomosse holds a first class BSc honours degree in Computer Science and a PhD from the Univ. of Bath. He is now there as a lecturer (asst. professor) and is part of the MTRC. His research interests lie in Computer Vision and Computer Graphics, specifically in the analysis of images and video to extract semantic representations of salient scene content and dynamics. Application areas focus on information retrieval and NPR.

Yi-Zhe Song received a first class Bsc in Computer Information Systems from the Univ. of Bath, UK, 2003; and the Diploma degree in Computer Science (with outstanding dissertation prize) from the Computer Laboratory, Univ. of Cambridge, in 2004. He is currently a PhD candidate in MTRC, Univ. of Bath, UK. His research interests include algorithms and applications in computer vision and graphics.

Peiyi Shen is a research officer and a PhD student in the MTRC, Computer Science at the Univ. of Bath. His research interests are in Computer Vision and Volume Visualization; the texture mapping and annotation of volume objects. He was with Agilent Technologies in the USA, UK, Malaysia and Singapore from 2000 to 2003, where he became a patent holder and a best performance (rank 1) employee. He was also a postdoctoral research fellow in the School of Computing at the National Univ. of Singapore in 2000.



Chuan Li is a PhD student in the Department of Computer Science, Univ. of Bath. He received his first degree in Software Engineering from Zhejiang Univ., PR China. His PhD study is supervised by Dr. Peter Hall, and his research focuses on the convergence of Computer Graphics and Computer Vision.