NAME

VCamera − camera model representation

SYNOPSIS

#include <vista/VCamera.h>

void VRegisterCameraType (void);

extern VRepnKind VCameraRepn;

VCamera *camera***;**

ld ... **-lvcam -lvista -lm -lgcc** ...

DESCRIPTION

Introduction

A Vista camera model represents the characteristics and position of a camera. By recording the camera's internal geometric and optical characteristics (intrinsic parameters) as well as its 3D position relative to a world coordinate system (extrinsic parameters), a camera model allows conversion between 3D world coordinates and 2D image coordinates.

Vista implements a camera model based on the work of Roger Tsai [1]. The model actually describes a combination of camera and frame grabber, and it accounts for three of the most significant sources of image distortion: radial distortion, displacement of the image center, and mismatch between camera and frame grabber horizontal scan rates.

Coordinate Systems

A Vista camera model describes the transformation from 3D world coordinates to 2D image coordinates in four steps via intermediate coordinate systems. All coordinate systems are right-handed with axes XY or XYZ. World, camera, and sensor coordinates are expressed in millimeters; image coordinates are in pixels. The four transformation steps are:

1. The 3D world coordinates of a point P, denoted *Pw*, are mapped to 3D camera coordinates *Pc* by a rotation *R* about the world origin followed by a translation *T*. The 3D camera coordinate system is centered at the camera's optical center (where optical rays converge in the absence of distortion). Viewed in the direction the camera is pointing, the camera coordinate system's +X axis points right, its +Y axis points down, and its Z axis lies along the optical axis. The transformation from world to camera coordinates is

   $Pc = R\ Pw + T$

2. The point *Pc* is projected onto the camera sensor as if the camera were an ideal pinhole camera. The focal length is *f* and the projected point, in an ideal undistorted sensor coordinate system, has coordinates *Psu* given by

   $Psu[0] = f * Pc[0]\ /\ Pc[2]$
   $Psu[1] = f * Pc[1]\ /\ Pc[2]$

3. The projected point is subjected to radial lens distortion, mapping it from undistorted sensor coordinates *Psu* to distorted sensor coordinates *Psd*. This distortion is modeled by a single factor proportional to the square of the distance from the optical axis to the projected point. This factor is parameterized by *k1*:

   $Psu[0] = Psd[0] * (1 + k1 * rr)$
   $Psu[1] = Psd[1] * (1 + k1 * rr)$

   where $rr = Psd[0] * Psd[0] + Psd[1] * Psd[1]$

4. The point is mapped from distorted sensor coordinates *Psd* to distorted image coordinates *Pid*. This mapping is determined by the image coordinates of the image center, *C*, the size of image pixels in sensor coordinates, *D*, and a factor *sx* accounting for mismatch between sensor and frame grabber horizontal scan rates. Unlike the sensor and camera, the image's coordinate system has its origin at the lower left corner of the image, its +X axis points right, and its +Y axis points up.

   $Pid[0] = C[0] + Psd[0]\ /\ D[0] * sx$

Pid[1] = *C*[1] - *Psd*[1] / *D*[1]

An additional coordinate system represents the image coordinates a point would have if the camera were to have no radial distortion. Undistorted image coordinates are denoted by *Piu*:

Piu[0] = *C*[0] + *Psu*[0] / *D*[0] * *sx*
Piu[1] = *C*[1] - *Psu*[1] / *D*[1]

A fuller description of this camera model and an associated calibration procedure may be found in [1].

Camera Model Components

A Vista camera model specifies the following information about a camera.

- the numbers of rows and columns of sensor elements
- the numbers of rows and columns of frame grabber pixels
- the dimensions of a sensor element
- the effective dimensions of a frame grabber pixel, *D*
- the image coordinates, *C*, of the intersection between the optical axis and the sensor
- a radial distortion coefficient, *k1*
- a horizontal scale factor, *sx*

Optionally, the camera model may also specify:

- the position of the camera as defined by a rotation *R* and translation *T*
- the camera's focal length, *f*
- any other attributes one may wish to associate with the camera, such as comments identifying the type of camera and frame grabber

The VCamera Type

A camera model is represented in memory by an object of type **VCamera**. Before using this type in your program (and before reading a **VCamera** object from a data file) you must register the type by calling **VRegisterCameraType**:

VRegisterCameraType ();

Following this call, the external variable **VCameraRepn** will contain the representation code assigned to identify the **VCamera** type (see **Vtype**(3Vi)).

Your program must be linked with the **vcam** library, as shown above in the SYNPOSIS section.

Creating, Copying, and Destroying Camera Models

The following routines create, copy, and destroy **VCamera** objects.

VCamera VCreateCamera (void);

VCreateCamera allocates memory for a camera model. All camera parameters are initialized to zero, and the camera model is given an empty attribute list.

VCamera VCopyCamera (VCamera *src***);**

VCopyCamera returns a copy of *src*. Included in the copy are both camera parameters and any other attributes associated with *src*.

void VDestroyCamera (VCamera *c***);**

VDestroyCamera releases the memory occupied by a camera model and its attributes.

Accessing Camera Model Parameters

The following table lists macros used to access fields of a **VCamera** object. Each macro takes a single argument of type **VCamera**, and accesses a value of the indicated type.

Macro	*Type*	*Description*

VCameraNSels	**VLong [2]**	numbers of columns and rows, respectively, of camera sensor elements
VCameraNPixels	**VLong [2]**	numbers of columns and rows of frame buffer pixels
VCameraSelSize	**VDouble [2]**	width and height of camera sensor elements, in millimeters
VCameraPixelSize	**VDouble [2]**	effective width and height of frame grabber pixels, *D*, in millimeters
VCameraImageCenter	**VDouble [2]**	the image coordinates, *C*, of the intersection between the optical axis and the sensor
VCameraRadialDistortion	**VDouble**	the radial distortion coefficient, *k1*
VCameraScaleX	**VDouble**	the horizontal scale factor, *sx*
VCameraExtrinsicsDefd	**VBoolean**	whether the camera model's extrinsic parameters are defined
VCameraRotation	**VDouble [3]**	the world-to-camera coordinate system rotation, represented as Euler angles denoting successive rotations about the world X, Y, and Z axes; each rotation angle is in radians, measured according to the right-hand convention (clockwise looking along the axis in the positive direction)
VCameraRotMatrix	**VDouble [3][3]**	the world-to-camera coordinate system, represented as a rotation matrix, *R*, for premultiplying location vectors
VCameraTranslation	**VDouble [3]**	the world-to-camera coordinate system translation, in millimeters
VCameraFocalLengthDefd	**VBoolean**	whether the camera model's focal length is defined
VCameraFocalLength	**VDouble**	the focal length, *f*, in millimeters
VCameraAttrList	**VAttrList**	a list of other attributes associated with the camera model

Each macro may be used as either an rvalue (e.g., on the right hand side of an assignment operator) or as an lvalue (on the left hand side of an assignment operator). See the EXAMPLES section, below, for an illustration of their use.

The Euler-angle and rotation-matrix representations of the world-to-camera rotation must be kept consistent. The **VSetCameraRotation**(3vi) routine can be used to ensure consistency.

Data File Representation

attribute-name: **camera {**
    nsels_x: *nsels*[0]
    nsels_y: *nsels*[1]
    npixels_x: *npixels*[0]
    npixels_y: *npixels*[1]
    sel_size_x: *sel_size*[0]
    sel_size_y: *sel_size*[1]
    pixel_size_x: *D*[0]
    pixel_size_y: *D*[1]
    image_center_x: *C*[0]
    image_center_y: *C*[1]
    radial_distortion: *k1*
    scale_x: *sx*
    rotation_x: *A*[0]

rotation_y: *A*[1]
rotation_z: *A*[2]
translation_x: *T*[0]
translation_y: *T*[1]
translation_z: *T*[2]
focal_length: *f*
other attributes

}

- The attributes within a **camera** object may be listed in any order.
- Distances, such as *sel_size*[0], are in millimeters; angles are in radians.
- The **npixels_x** and **npixels_y** attributes are optional; if omitted, their values default to

  npixels[0] = 512
  npixels[1] = 480

- The **pixel_size_x** and **pixel_size_y** attributes are optional; if omitted, their values default to

  D[0] = *sel_size*[0] * *nsels*[0] / *npixels*[0]
  D[1] = *sel_size*[1] * *nsels*[1] / *npixels*[1]

- The **image_center_x** and **image_center_y** attributes are optional; if omitted, their values default to

  C[0] = *npixels*[0] / 2
  C[1] = *npixels*[1] / 2

- The **radial_distortion** attribute is optional; if omitted, its value defaults to 0.
- The **scale_x** attribute is optional; if omitted, its value defaults to 1.
- The attributes **rotation_x** through **translation_z** may be omitted, indicating that the extrinsic parameters are undefined.
- The **rotation_x**, **rotation_y**, and **rotation_z** attributes describe the world-to-camera coordinate system rotation as successive rotations about the world X, Y, and Z axes; each rotation angle is in radians, measured according to the right-hand convention (clockwise looking along the axis in the positive direction).
- The **focal_length** attribute may be omitted, indicating that the focal length is undefined.

Camera Calibration

Given a series of data points whose positions are known in both world coordinates and distorted image coordinates, several of a camera model's parameters can be estimated. Using **vcamcal**(1Vi) or **VCalibrateCamera**(3Vi), you can

- estimate both intrinsic parameters (*C*, *D*, *k1*, *sx*, and *f*) and extrinsic ones (*R* and *T*);
- estimate extrinsic parameters only, retaining the existing values of intrinsic ones; or
- measure the accuracy of an existing camera model.

The estimation procedure differs slightly depending on whether the data points lie in a single world plane.

- Coplanar points must all have a Z world coordinate of zero. In addition, for numerical stability, all points should lie some distance from the origin of the world coordinate system. The plane containing the points should not be parallel to the imaging plane; a relative angle of 30 degrees is recommended. The horizontal scale factor, *sx*, cannot be estimated from coplanar points.
- Noncoplanar points must not lie on a single world plane. The horizontal scale factor, *sx*, is among the parameters that can be estimated.

Various programs produce or accept lists of data points in Vista data files. A list of data points is represented as an attribute list with each data point occupying a single attribute. Data point values are, in turn, attribute lists of coordinate values. This is best illustrated by an example, here showing a list of (only) two data points:

points: {

```
        0: {
                w_x: 10
                w_y: 10
                w_z: 30
                i_x: 193.45
                i_y: 185.20
        }
        1: {
                w_x: 10
                w_y: 30
                w_z: 30
                i_x: 192.68
                i_y: 148.59
        }
}
```

The **w_x**, **w_y**, and **w_z** attributes record the world coordinates of a data point; **i_x** and **i_y** record its image coordinates. The names of the data points (‘‘0’’ and ‘‘1’’ in the example) are not significant.

EXAMPLES

This Vista data file contains a minimal, uncalibrated camera model:

```
V-data 2 {
        camera1: camera {
                camera_name: "Sony XC-77RR"
                nsels_x: 768
                nsels_y: 493
                sel_size_x: 0.011
                sel_size_y: 0.013
        }
}
^L
```

The number of rows and columns of frame grabber pixels for the camera model *c* may be set to 480 and 512, respectively, by:

```
VCameraNPixels(c)[0] = 512;
VCameraNPixels(c)[1] = 480;
```

Its focal length may be set by:

```
VCameraFocalLength(c) = 30.0;
VCameraFocalLengthDefd(c) = TRUE;
```

SEE ALSO

vcalsyn(1Vi), **vcamcal**(1Vi),
Vattribute(3Vi), **Vfile**(5Vi), **Vlib**(7Vi)

AUTHORS

Calibration procedure by Roger Tsai [1]. Original implementation by Reg Willson <rgw@cs.cmu.edu>. Vista implementation by Art Pope <pope@cs.ubc.ca>.

REFERENCES

[1] R. Tsai, ‘‘A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses’’, **IEEE J. Robotics and Automation** (1987), pp. 323-344.

[2] J. Weng, P. Cohen, M. Herniou, ‘‘Camera calibration with distortion models and accuracy evaluation’’, **IEEE Trans. Pattern Analysis and Machine Intelligence** (1992), pp. 965-980.

LIST OF ROUTINES

The following table summarizes other Vista library routines that operate on camera models. Each routine is documented elsewhere by a section 3Vi man page named for the routine.

For converting among coordinate systems:

VWorldToCameraCoords

VCameraToWorldCoords

VWorldToImageCoords

VImageToWorldCoords

VDistortedToUndistortedImageCoords

VUndistortedToDistortedImageCoords

VDistortedToUndistortedSensorCoords

VUndistortedToDistortedSensorCoords

VImageToUndistortedSensorCoords

For evaluating camera model accuracy:

VNormalizedCameraCalibrationError	computes statistics of Weng et al.'s normalized calibration error [2]
VObjectSpaceCameraCalibrationError	computes statistics of error in world coordinates

Miscellaneous:

VCalibrateCamera	for estimating camera model parameters from calibrated data points
VSetCameraRotation	for setting Euler angles and rotation matrix consistently