A calibrated 3D dataset for automatic evaluation of keypoint detectors and descriptors

P. Bendale*, J. Cameron*, B. Triggs† and
N. Kingsbury*

*Signal Processing Laboratory, Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK
† Laboratoire Jean Kuntzmann, BP 53, 38041 Grenoble Cedex 9 France

CUED/F-INFENG/TR. 655

1 Introduction

Previous evaluations of keypoint detectors and descriptors have tended to focus on planar scenes, relying on homographies to generate the ground truth [6, 8]. Evaluations on 3D scenes include [4] (using trifocal tensors on natural scenes) and [7] (using calibrated images from a turntable). In order to provide an accurate evaluation of methods on real scenes, we wanted to use a method similar to [7]. Unfortunately, although the authors provide a robust framework, they have not made their complete ground truth, calibration and test software publicly available. Moreover, the objects typically only occupy a small portion of their images and there are typically many detections on the image backgrounds, which are neither plain nor realistic. To overcome these problems, we introduced a new dataset, 'Cambridge toy cars dataset' in [2]. This report describes the dataset in further detail.

2 Cambridge toy cars dataset

This dataset contains 4000 calibrated images from two cameras of 40 different toy vehicles on a turntable and a plain grey background. Some example images are shown in Figure 1. There are 51 images (taken in steps of 3.6°) per car per camera spanning 180°. The images are available in PNG and raw (NEF) format. The calibration images and camera parameters including lens distortions, and the MATLAB test scripts are available for download. We used the publicly available DLR CalDe/CalLab toolbox [9] for calibration and DCRaw [3] for format conversion (including debayering) of the NEF (Nikon raw format) files. For convenience, the data is split into three sets (each of which has different calibration parameters), containing 10-15 car sequences. Please note that these images are quite big - each PNG is 3039×2014 (5.5MB) and each NEF file is 3039×2014 (17MB). It is possible to download each set separately. We provide gamma-compressed PNG files in the standard download. RAW files without gamma compression are also available.

Geometrically, one camera ('auxiliary') is above the other ('reference/test') and somewhat in front of it, so that both cameras are at approximately the same distance to the centre of the turntable (and looking directly at it with an angular separation of 15°). This arrangement produces near vertical epipolar lines between the reference and auxiliary images, near horizontal ones between adjacent reference/test images, and similar image scales in all (which simplifies evaluations of keypoint scale estimates). In practice, for each 'reference/test' image in turn ('reference'), we use epipolar geometry against its corresponding 'auxiliary' image to find possible matches, then use 3-image epipolar geometry to evaluate whether a corresponding point was found in the desired 'test' image (the 'reference/test' one at a given angular separation from the current 'reference').

2.1 Camera calibration matrix

This matrix contains information about the principal point (x_0, y_0) in the image, the skew factor s and the focal length in x and y directions. If a camera does not have equal scale factors in x and y directions, *i.e.* has non-square pixels, then the focal length gets scaled unequally in the two directions. If f is the focal length and m_x and m_y are the scale factors in x and y directions, then $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent the effective focal length in x and y directions respectively. The camera calibration matrix K in it's general form is given by

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}. \quad (1)$$

The rotation and translation matrices together comprise the external calibration of a camera. The complete camera projection matrix P can be expressed as

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{T}]. \quad (2)$$

The Fundamental matrix $\mathbf{F}_{21}$, that projects points from image I_1 to image I_2 can be obtained once the projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$ are known. The projection matrices of the test and reference images in our 3-image setup are related by a pure rotation matrix. These definitions and further details are available in [5].

Figure 1: Central 1024 tall ×1536 wide patch from each image in the dataset.

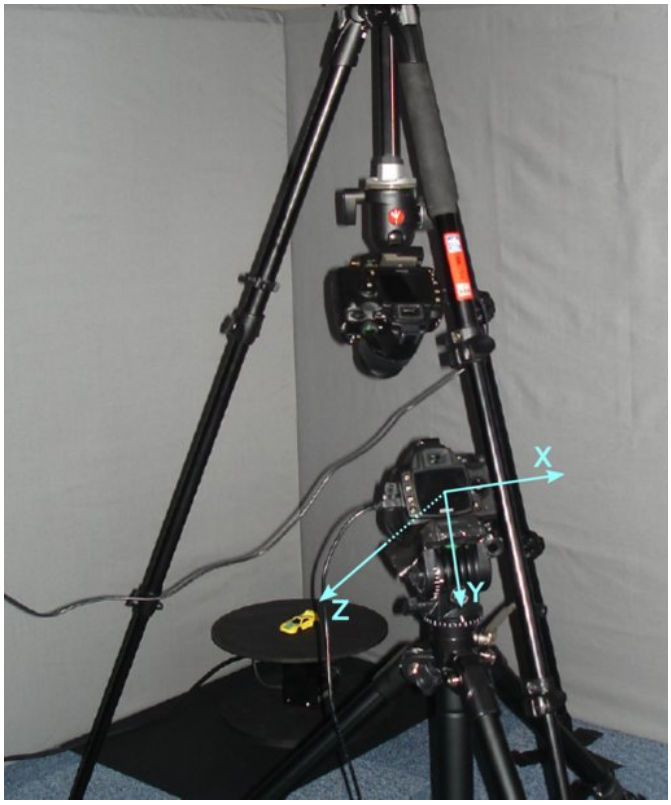


Figure 2: The rig used to capture the dataset. Note that the upper camera is inverted. All images from the upper camera are inverted, but no correction or de-rotation is necessary to handle this. All software provided assumes that the upper camera is inverted. The calibration process handles this correctly as the calibration images from upper camera are also inverted. If desired, the images may be de-rotated (by 180°)for display purposes.

2.2 Radial Distortion

The radial distortion model used in DLR CalDe/CalLab toolbox [9] (which has been used for the calibration of our dataset) can be expressed as

$$\begin{aligned} x' &= x + k_1\,(x - x_0)\,\hat{r}^2 + k_2\,(x - x_0)\,\hat{r}^4 + \dots \\ y' &= y + k_1\,(y - y_0)\,\hat{r}^2 + k_2\,(y - y_0)\,\hat{r}^4 + \dots \end{aligned} \tag{3}$$

where k_1 and k_2 are radial distortion coefficients, $\hat{r}$ is the observed radial distance of a point measured as

$$\hat{r}^2 = \left(\frac{(x - x_0) - s\,(y - y_0)/f_y}{f_x}\right)^2 + \left(\frac{y - y_0}{f_y}\right)^2 \tag{4}$$

and (x', y') are the predicted perspective projections of (x, y). The distortion is due to the camera lens so it happens after perspective projection but before digitizing. The distortion is due to the camera lens so it happens after perspective projection but before scaling and digitizing.

While converting from actual image positions to ideal image positions, it is necessary to undo the effect of lens distortion (*i.e.* undistort the coordinates), whereas while converting from ideal image positions to actual image positions, it is necessary to apply appropriate lens distortion (*i.e.* distort the coordinates). Ideal image positions obey all rules of linear projection (*i.e.* Epipolar lines are straight lines *etc.*). Lens distortion has the effect of converting these into curves diverging away from the optical centre in the actual image positions. Lines in the real world do not get imaged as lines. The effect is not negligible if one wishes to establish sub-pixel correspondences.

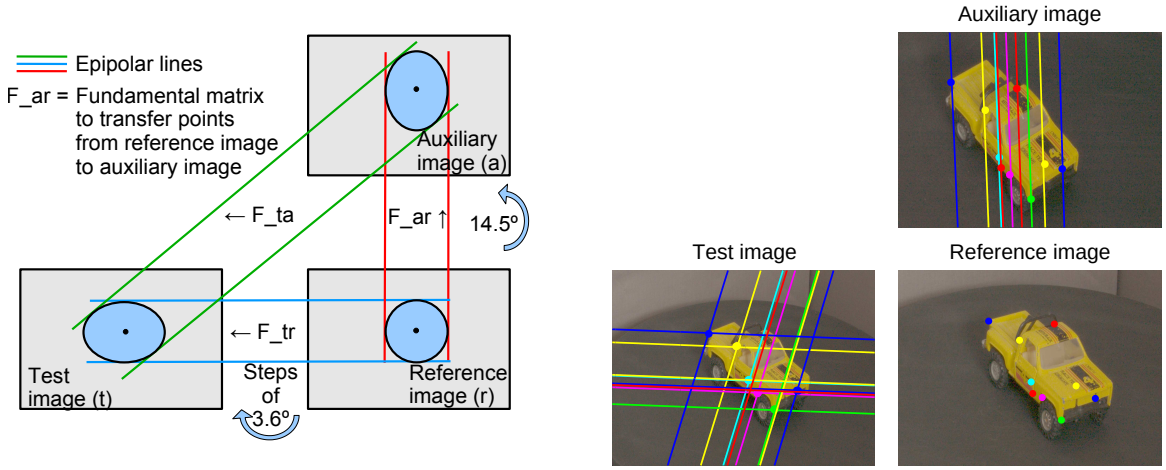


Figure 3: The set-up used for our dataset and experiments is similar to [7]. Left: The epipolar matching scheme. For each keypoint in the reference image, epipolar constraints and normalised colour cross-correlation are used to estimate the location of the corresponding keypoint in the auxiliary image. The intersection of the reference-image and auxiliary-image epipolar lines in the test image then gives us the predicted location of the corresponding keypoint in the test image (if any). Right: Some example images with epipolar lines.

3 Calibration data

The calibration images used in the calibration process and the corners detected in those images are available at `http://www-sigproc.eng.cam.ac.uk/imu`. A few examples of calibration images are shown in Figure 4. DLR CalDe/CalLab toolbox was used for the calibration.

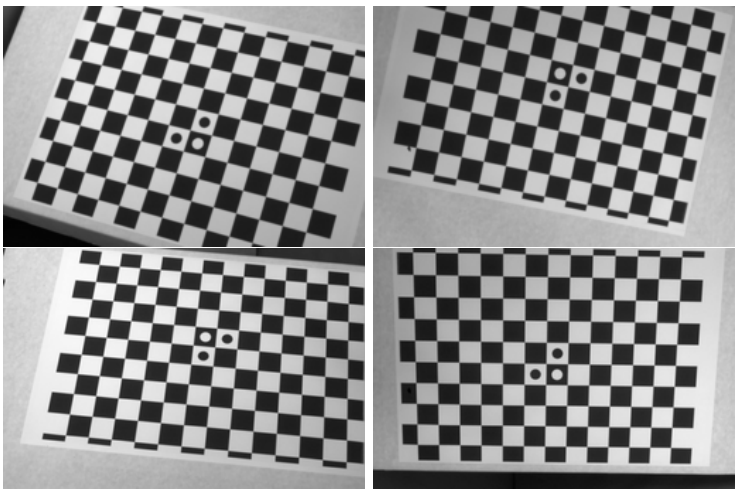


Figure 4: Example calibration images from sequence 1. We used the A4 size calibration pattern from DLR CalDe/CalLab toolbox. The circles in the centre of the calibration pattern are useful for recovering the (x, y) axis with respect to the origin (including the sign). The z-axis is perpendicular to the calibration plane and always points inside the pattern.

Each archive seqM.zip contains (M is the sequence number and NN are calibration image numbers)

1. Images of the calibration pattern, named NN.lower.cal.png and NN.upper.cal.png (8-10 each)
2. Coordinates of corners detected in these images, named NN.pts (8-10 each)
3. Coordinates of corners used in the calibration process, named shotNN.pts (8-10 each)
4. Calibration output, named seqM_hh.par (1 each)

Calibration output from seq1_hh.par, seq2_hh.par and seq3_hh.par and information about the position of the turntable with respect to the cameras is available in loadCalibrationParams.m (see Sec. 4).

A tutorial on how to use the DLR CalDe/CalLab toolbox for calibration is available at [9]. CalLab usage help and explanation of calibration process can be found in Documentation.txt and explanation of lens distortion model used in the calibration process can be found in Distortions.txt, both of which are also available at [9].

4 Ground truth

The ground truth is provided in a file named loadCalibrationParams.m. This file contains information about the following parameters:

Term	Explanation
upper.A	Camera calibration matrix($\mathbf{K}$) for upper camera
lower.A	Camera calibration matrix($\mathbf{K}$) for lower camera
R, T	Rotation and translation of upper camera relative to centre of lower camera
upper.k1, upper.k2	Radial distortion parameters for upper camera
lower.k1, lower.k2	Radial distortion parameters for lower camera
Rc, Tc	Rotation and translation of lower camera relative to centre of lower camera
video.width	Image width
video.height	Image height
X0	Centre of turntable in world coordinates
u	Unit vector in the direction of axis of rotation of turntable

loadCalibrationParams.m

```
% loadCalibrationParams.m

% This function provides the rotation matrix, translation and
% internal camera parameters for the two cameras.
% Lower camera is the reference camera, R and T are the transformation
% from the centre of the lower camera to the centre of the upper camera.
% The world coordinates of the centre of the turntable (X0)
% and unit vector corresponding to the axis of rotation of the turntable (u)
% are retrieved by fitting a (least squares) conic to the world coordinates
% of points marked on the edge of the turntable.
% These world coordinates are obtained by
% triangulation of the corresponding points marked on a set of
% reference (lower) and auxiliary (upper) images.
% Coordinate convention: Top left corner of the image is (x = 1,y = 1)
% The images are 3039 wide (x) and 2014 tall (y)
% Note that R and T are not in standard format, hence need to be transposed
% before using to form Pref or Paux

function [upper,lower,video,Pref,Paux,X0,u,R,T] = loadCalibrationParams(setNum)
% Sequence 1
% 01,02,04,05,06,07,08,09,10,12,27,28,47,59
% Sequence 2
% 15,18,21,23,29,31,32,33,41,44,45,46,49,50,51,61
% Sequence 3
% 11,16,24,25,36,48,53,57,68

if setNum == 1
    % Sequence 1
    upper.A = [ 6731.51   8.09608   1548.91;
                0.000000  6732.26   1019.14;
                0.000000  0.000000  1.00000];
    upper.k2 = 0.104508;
```

```
    upper.k1 = -0.0762281;
    T = [ -8.54774 -204.596 36.7655];
    lower.A = [ 6725.32   6.13373   1502.66;
                0.000000  6731.29   1084.56;
                0.000000  0.000000  1.00000];
    R = [ -0.995878   0.0893587  0.0155455;
          -0.0832026  -0.968267  0.235659;
           0.0361104  0.233395   0.971711];
    lower.k2 = -0.969736;
    lower.k1 = -0.0238495;
    video.width = 3039;
    video.height = 2014;
    Rc=[ 1  0  0;
         0  1  0;
         0  0  1];
    Tc=[ 0  0  0];
    Pref = lower.A * [Rc' Tc'];
    Paux = upper.A * [R.' T.'];
    X0 = [25.6262; 0.2633; 811.0786];
    u =  [0.0358; -0.8801;  -0.4733];
end
if setNum == 2
    % Sequence 2
    upper.A=[ 6804.19   10.4926   1579.23;
              0.000000  6822.17   927.123;
              0.000000  0.000000  1.00000];
    upper.k2= -0.106142;
    upper.k1= -0.0859228;
    T=[ -9.31635 -209.800 37.6310];
    lower.A=[ 6830.58   5.62836   1479.75;
              0.000000  6875.41   1181.99;
              0.000000  0.000000  1.00000];
    R=[ -0.995903   0.0895495  0.0126142;
        -0.0837961  -0.966238  0.243646;
         0.0340066  0.241590   0.969782];
    lower.k2= -0.0776615;
    lower.k1= -0.105737;
    video.width= 3039;
    video.height= 2014;
    Rc=[ 1  0  0;
         0  1  0;
         0  0  1];
    Tc=[ 0  0  0];
    Pref = lower.A * [Rc' Tc'];
    Paux = upper.A * [R.' T.'];
    X0 = [33.7781; -16.9800; 819.8550];
    u =  [0.0290;  -0.8731;   -0.4867];
end
if setNum == 3
    % Sequence 3
    upper.A=[ 6684.47   5.63880   1563.36;
              0.000000  6688.31   986.381;
              0.000000  0.000000  1.00000];
    upper.k2= 0.519152;
    upper.k1= -0.0823516;
    T=[ -8.31937 -200.730 36.7853];
    lower.A=[ 6671.42   7.92804   1497.92;
              0.000000  6670.73   1110.27;
              0.000000  0.000000  1.00000];
    R=[ -0.995853    0.0899773  0.0134746;
        -0.0841423   -0.967178  0.239766;
         0.0346058   0.237638   0.970737];
    lower.k2= -0.985570;
```

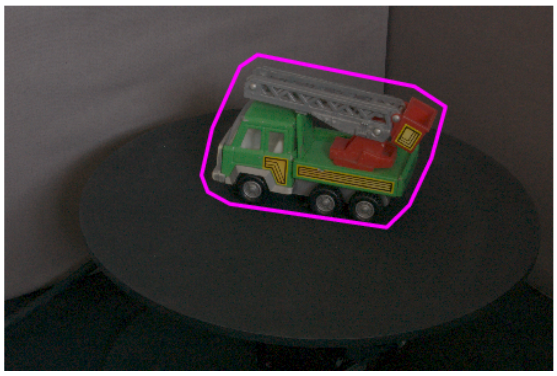


Figure 5: An example of the polygonal boundary superimposed on an image of a car.

```
    lower.k1= -0.00243372;
    video.width= 3039;
    video.height= 2014;
    Rc=[ 1  0  0;
         0  1  0;
         0  0  1];
    Tc=[ 0  0  0];
    Pref = lower.A * [Rc' Tc'];
    Paux = upper.A * [R.' T.'];
    X0 = [28.5223; -3.3355; 799.2905];
    u =  [-0.0243;  0.8776;   0.4788];
end
end % function end
```

5 Polygonal boundaries

Although not needed for evaluation of keypoints, convex polygon boundaries for 15 sequences of the cars (for both cameras) are also available. A demo is available at `http://www-sigproc.eng.cam.ac.uk/imu`. An archive named convhull.zip contains files named convhullCCangAAAAlower.mat and convhullCCangAAAAupper.mat. Here CC is car number and AAAA is angle number. Each file contains the coordinates of the convex hull of the polygonal boundary of the toy car in (x, y) format with top-left of image being $(1, 1)$.

The following MATLAB commands may be used to see the convex hull on the toy car

```
K = load('convhull01ang0lower.mat'); % will load into K.chull
img = imread('car01ang0lower.png');
imagesc(img); % patch uses current figure for plotting on
hold on; plot(K.chull(:,1),K.chull(:,2),'m','linewidth',2); hold off;
```

6 Data capture process

The images were captured using two Nikon D40 cameras (with standard 18–55mm lens), a pan-tilt unit (PTUD46 from Directed Perception) on a Linux system. The pan-tilt unit handles the rotation of the turntable. Both the pan-tilt unit and the cameras are controlled remotely using a shell script or the command prompt. More information about related commands may be seen in gphoto2 command reference and PTU D46 user manual at [1]. The shell script captureCar.sh that was used to capture data (*i.e.* interact with PTUD46 and the cameras) is reproduced here:

captureCar.sh

```
# Arguments for captureCar.sh: $1 = carNumber
# Run using the following command
# ./captureCar.sh 01
# where 01 is the first argument, specifying the carNumber

MIN=-1750        # First position at -180 deg
MAX=1750         # Last position at 180 deg
INTERVAL=70      # pan unit step size, 70 positions = 3.6 deg
SPEED=100        # speed of rotation of pan tilt unit in positions/second
DELAY=1
PTUFILE=/dev/ttyS0 # Port where pan tilt unit is connected

# Read this off the list of ports
# This changes every time the cameras are plugged in
USB1=usb:001,005 # Port where lower camera is connected
USB2=usb:001,006 # Port where upper camera is connected
CAMERANAME="Nikon DSC D40 (PTP mode)"
FOLDER1=/store_00010001/DCIM/100NCD40
FOLDER2=/store_00010001/DCIM/100NCD40

# Save images to memory card
# In recent versions there is a --capture-image-and-download option as well
# for some cameras but make sure capturetarget=1 is set
# to ensure automatic increment of the filecounter
gphoto2 --camera=$CAMERANAME --port=$USB1
        --set-config /main/settings/capturetarget=1
gphoto2 --camera=$CAMERANAME --port=$USB2
        --set-config /main/settings/capturetarget=1
# Set the aperture and exposure time
gphoto2 --camera=$CAMERANAME --port=$USB1
        --set-config /main/capturesettings/f-number=15
gphoto2 --camera=$CAMERANAME --port=$USB2
        --set-config /main/capturesettings/f-number=15
gphoto2 --camera=$CAMERANAME --port=$USB1
        --set-config /main/capturesettings/exptime=30
gphoto2 --camera=$CAMERANAME --port=$USB2
        --set-config /main/capturesettings/exptime=30

echo "Starting up"
# Reset pan axis only
echo 'rp' > $PTUFILE
# Set pan speed
echo "ps$SPEED" > $PTUFILE
# Move to the desired directory
cd /home/dataset/data
# Set absolute pan position to the first position
echo "pp$MIN" > $PTUFILE
# A big enough sleep to allow the pan tilt unit
# to change to the desired pan position
sleep 60
# Within the for loop, move $INTERVAL steps at a time,
# Capture an image from both cameras
# until the last image is captured
for i in `seq $MIN $INTERVAL $MAX`;
do
    # Move to absolute pan position = $i
    echo "pp$i" > $PTUFILE
    # Display current pan position on the command prompt
    echo "pp$i"
    # Allow time for the pan-tilt unit to stabilise
    sleep $DELAY
```

```
    # Specify file name and store as NEF(raw)
    # Capture from lower camera
    gphoto2 --camera=$CAMERANAME --port=$USB1
            --filename=car$1ang${i}lower.nef --capture-image --get-raw-data=2
    # Capture from upper camera
    gphoto2 --camera=$CAMERANAME --port=$USB2
            --filename=car$1ang${i}upper.nef --capture-image --get-raw-data=2
done

# To see photos on camera, use the following command
# gphoto2 --list-files

# Explicitly delete all images on the memory card to free up space
gphoto2 --camera=$CAMERANAME --port=$USB1 -D --folder=$FOLDER1
gphoto2 --camera=$CAMERANAME --port=$USB2 -D --folder=$FOLDER2
```

7 Sample code

A few (fairly basic) example routines illustrating the use of the dataset and the associated ground truth are available on the website. We provide a short description of each function here.

distort_.m This function induces lens distortion. This is needed while transferring points from ideal image positions to actual image positions.

undistort_.m This function removes lens distortion. This is used while transferring points from actual image positions to ideal image positions. This function is part of the DLR CalDe/CalLab toolbox [9].

example_transfer_ar.m Example of point transfer from reference image to auxiliary image, *i.e.* from lower camera to upper camera for the same view. This example produces Figure 6(a).

example_transfer_tr.m Example of point transfer from reference image to test image, *i.e.* from lower camera at one view to lower camera at a different view. This example produces Figure 6(b).

example_transfer_ta.m Example of point transfer from auxiliary image to test image, *i.e.* from upper camera at one view to lower camera at a different view. This example produces Figure 6(c).

loadCalibrationParams.m This function provides the ground truth.

Fmat_from_Ps.m This function computes a Fundamental matrix $\mathbf{F}$ for a pair of images from their projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$. This function is a part of [10].

getRotationAboutAxis.m This function computes the Rotation matrix $\mathbf{R}$ and Translation matrix $\mathbf{T}$ between two views for a given camera (usually the lower camera) using information about the angular separation between the two views, the centre of rotation and axis of rotation.

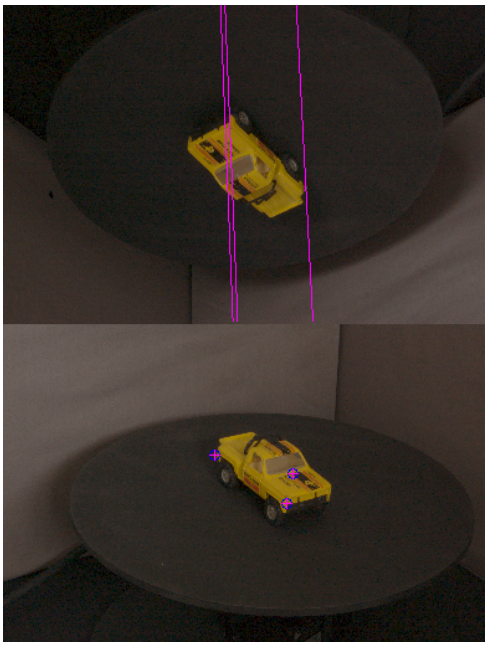
captureCar.sh The shell script used to capture the dataset. This includes software for controlling the pan-tilt unit and the two cameras.

References

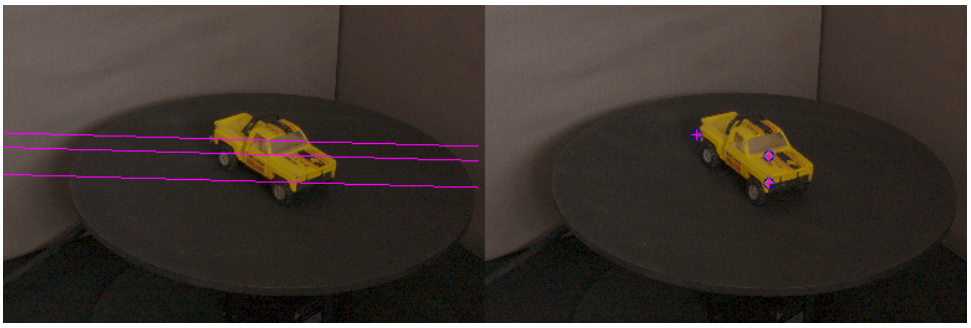
[1] Directed perception PTU D46. User manual and command reference available at `http://www.dperception.com/pdf/products/PTU-D46/PTU-manual-d46.pdf`. 8

[2] P. Bendale, B. Triggs, and N. Kingsbury. Multiscale keypoint analysis based on complex wavelets. In *British Machine Vision Conference*, 2010. 2

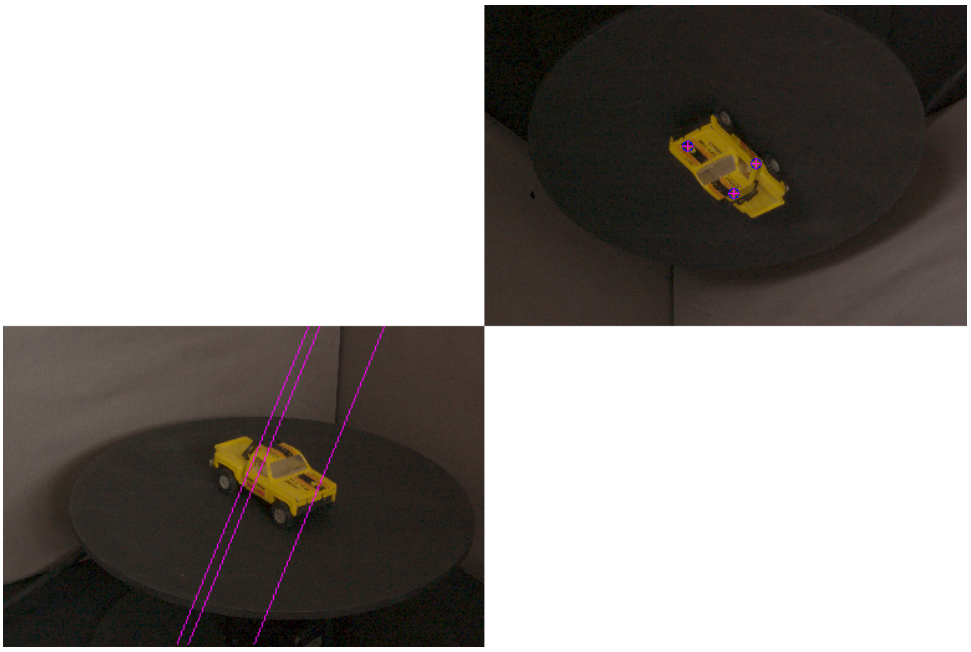
[3] D. Coffin. DCRAW: Decoding raw digital photos in linux. Available at `http://cybercom.net/~dcoffin/dcraw`, 2008. 2



(a) Example use of $\mathbf{F}_{ar}$



(b) Example use of $\mathbf{F}_{tr}$



(c) Example use of $\mathbf{F}_{ta}$

Figure 6: Point transfer examples between different views.

[4] F. Fraundorfer and H. Bischof. A novel performance evaluation method of local detectors on non-planar scenes. In *IEEE Conf. Computer Vision & Pattern Recognition - Workshops*, volume 03, page 33, 2005. 2

[5] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 2

[6] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005. 2

[7] P. Moreels and P. Perona. Evaluation of features detectors and descriptors based on 3D objects. *Int. J. Computer Vision*, 73(3):263–287, 2007. 2, 5

[8] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997. 2

[9] K. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. DLR CalDe and DLR CalLab. Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Oberpfaffenhofen, Germany. Available at `http://www.robotic.dlr.de/callab/`. 2, 4, 6, 10

[10] Bill Triggs. Assorted camera pose routines. Available at `http://ljk.imag.fr/membres/Bill.Triggs/src/pose.tar.gz`. 10