# Questions: 1.3, 1.4, 2.4, 2.5

Candidate number: H801L
Word count: 2399

15th November 2020

# 1 Q 1.3

## 1.1 Detectors

- Edge detection - D. Marr, 1980 [1]
- Corner detection - C. G. Harris, 1987 [3]
- Blob detection - D.G. Lowe, 2004 [4]

**Edge detection - D. Marr, 1980** Algorithm presented by Marr [1] convolves a second derivative of Gaussian 2D filter with an image (Eq. 1) to detect spatial pixel intensity changes. Edges are detected at zero-crossing of resultant function. It is expected that these will correspond with intensity changes in: illumination, reflectance, surface orientation and distance from the viewer.

$$\nabla^2 G(x, y) * I(x, y) \tag{1}$$

However, before applying $\nabla^2 G(x, y)$ filter, we would first apply a Gaussian filter of various $\sigma$ values, to obtain several channels at different resolutions. By the spatial coincidence assumption, we believed that physical edges will have coincident zero-crossings in channels of similar resolutions. Those zero-crossings which are not consistent across channels are disregarded.

**Corner detection - C. G. Harris, 1987** Any spatial perturbation to a patch centered at a corner would yield an intensity change to that patch. The intensity change can be tracked using Morave'c function (Equation 3) [2] and forms a basis for Harris [3] corner detection.

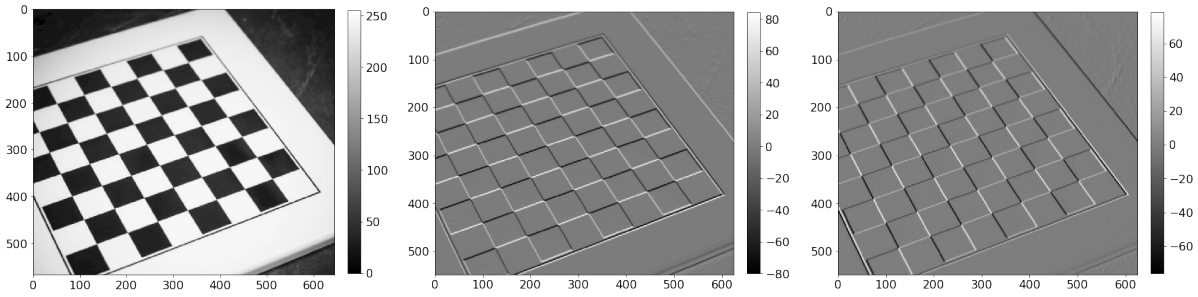
$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2 \tag{2}$$

We start, by removing noise with Gaussian filter (Equation 3) and computing intensity derivatives in $x$ and $y$ directions (Figure 1.a and 1.b). Then approximate Equation 3, with its first order Taylor expansion approximation, for computational reasons (Equation 4). Next form a Gaussian patch by convolving entries of matrix $M$ entries with Gaussian filter $\nabla G(x, y)$.

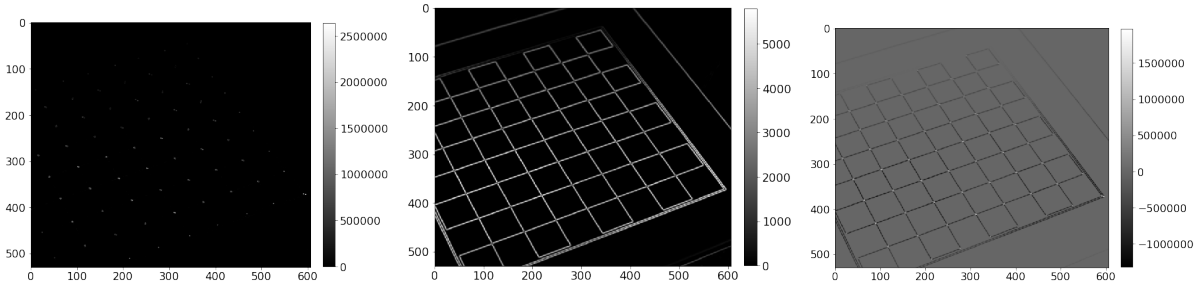
$$\nabla G(x, y) * I(x, y) \tag{3}$$

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{4}$$

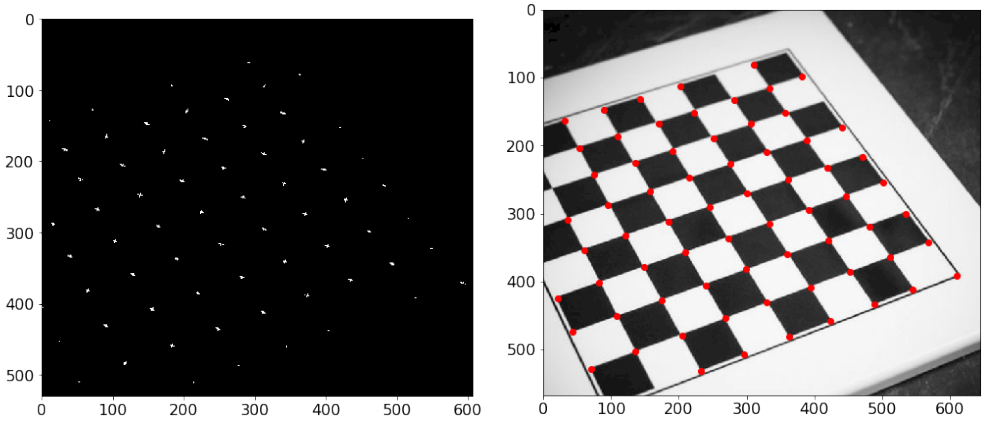
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{5}$$



(a) chessboard in grey scale (b) $I_y$ (c) $I_x$



(d) $det(M) = I_{x2}I_{y2} - I_{xy}^2$ (e) $trace(M) = I_{x2} + I_{y2}$ (f) $R = det(M) - k(trace(M)^2)$



(g) pixels with $R > 20000$ (h) Corners after Non-max suppress.

Figure 1

To identify corners we need to look at eigenvalues of matrix $M$. A pixel is a corner when both eigenvalues are large. To avoid computing matrix inverse we can find indirectly by computing corner response $R$ (Figure 1.f). Patches for which $R$ is larger than the threshold hyperparameter (here $tr = 20000$), correspond to matrix $M$ with two large eigenvalues. In a final step we apply non maximum suppression to assign corners to pixels with highers $R$ value in the neighborhood (Figure 1.h).

$$R = det(M) - k(trace(M))^2 \tag{6}$$

**Blob detection - D. G. Lowe, 2004** We convolve image with a Laplacian of a Gaussian filter, which takes high values when convolved with a blob of a corresponding size. Size of

the blob detected depends on the sigma of the detector used, thus we use filters of different sizes to detect blobs across a wide scale space. Out of practical consideration we decide to smooth image with Gaussians of sizes as defined by Eq. 7, which means that every $s$ times, the filter size will double.

$$\sigma_i = 2^{\frac{i}{s}}\sigma_0 \tag{7}$$

Instead of doubling size of a filter we can decrease the resolution of an image by a factor of 2 via sub-sampling. This will give the same effect at a lower cost. Additionally, application of incremental Gaussian blurs, allows to reuse smoothed images and obtain further smoothing at lower cost (i.e. convolving smoothed image with $G(\sigma_2)$ is less expensive then an original image with filter $G(\sqrt{\sigma_1^2 + \sigma_2^2})$)

$$G(\sigma_1) * G(\sigma_2) = G(\sqrt{\sigma_1^2 + \sigma_2^2}) \tag{8}$$

Final computational saving involves replacing of LoG with Difference of Gaussians (DoG). This simplification relies on an approximation presented in the Equation 9.

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G(x, y, \sigma) \tag{9}$$

## 1.2 Applying detectors for calibration

To perform camera calibration we need to localise features on the image plane ($\tilde{\boldsymbol{w}}$), which correspond to known locations in the world coordinate system ($\tilde{\boldsymbol{X}}$). We use these to estimate matrix $P$ (Equation 10) which defines the mapping between the two coordinate systems.

$$\tilde{\boldsymbol{w}} = P\tilde{\boldsymbol{X}} \tag{10}$$

**Edge detector** becomes activated where pixel intensities change on the image. It outputs curves, which makes it a poor choice for point-based calibration. A better approach is to optimise the alignment of a known pattern to the contours (edges) of the image, by fitting the projection matrix $P$ [5].

Corners have well-defined positions, which makes **Corner detector** suitable for point-based camera calibration. Corners are not invariant to scaling. Therefore it is important for the calibration pattern to have distinctive edge intersections such as those on a chessboard.

Scale invariance property of **Blob detector** makes it more versatile than corner detector. We match blobs of correct sizes with regions of uniform intensity on the image, and use their center of gravity as feature points. Although spatial regions are sensitive to perspective distortion, we can account for that during calibration (see Q1.4).

Accurate localisation of features is important, however we also need a way to pair features with their corresponding world locations. For point-based calibration, we would use an asymmetric grid calibration pattern (chessboard, circle-grid) to reject features which don't match the grid structure and make the correspondence matching easier for those which do. We would usually need to consider multiple views of a calibration pattern for a higher accuracy, when using features of lower stability such as corners.

# 2 Q 1.4

## 2.1 Perspective Camera Model

Perspective camera model can be defined as a series of transformations from a 3D world coordinates ($\tilde{\boldsymbol{X}}$) onto 2D image plane ($\tilde{\boldsymbol{w}}$), as presented in the Equation 11. In consists of a Camera matrix $\mathbf{K}$, rotation matrix $\mathbf{R}$, and translation vector $T$ [6].

$$\tilde{\boldsymbol{w}} = \mathbf{K}[\mathbf{R}|T]\tilde{\boldsymbol{X}} \tag{11}$$

**Rotation matrix** specifies required rotation of object's coordinate system to align it with image plane, then **Translation vector** defines the shift of the World coordinates' origin to match that of the image plane. $[\mathbf{R}|T]$ has total of 6 unkown parameters, 3 for translation and 3 for rotation (9 components of $\mathbf{R}$ can be specified by 3 angles of rotation, Equation 13). These two transformations are presented in a form a single $3x4$ matrix, because we use homogeneous coordinates.

$$[\mathbf{R}|T] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_x \\ r_{21} & r_{22} & r_{23} & T_y \\ r_{31} & r_{32} & r_{33} & T_z \end{bmatrix} \tag{12}$$

$$R = R_x(\psi) \times R_y(\varphi) \times R_z(\theta) \tag{13}$$

$\mathbf{R}$ and $T$ are called extrinsic parameters because they change between camera views. **Camera matrix** (Equation 14) is another transformation and accounts for intrinsic parameters of a camera (view invariant).

$$\mathbf{K} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{14}$$

Focal length $f$ is the distance from a lens to CCD and specifies the scaling of object's on to image plane. A camera has a single focal length $f$, however because CCD's pixels can be rectangular, we account for that using a scaling factors for each dimension, $k_u$ and $k_v$. Parameters $u_0$ and $v_0$ define CCD's offset from optical axis which arises during CCD assembly, as limited accuracy of humans will make CCD minimally shifted.

## 2.2 Camera calibration with 3D object

Direct estimation of 10 unknown parameters of Perspective Camera Model is complicated due to non-linear constraints of the model. However, we can use a more general **projective camera model**, displayed in Equation 15, which is easier to solve.

$$\tilde{\boldsymbol{w}} = \mathrm{P}\tilde{\boldsymbol{X}} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tilde{\boldsymbol{X}} \tag{15}$$

To solve for these $3x4$ matrix we need to collect a set of 6 feature points, using techniques defined in Q1.3. These will allow us to form a set of linear equations by substituting them for $\tilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{X}}$. We need 6 points $(x, y)$, because we need to form at least 11 simultaneous equations, as presented in Equation 16.

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = 0 \quad (16)$$

This can be solved with Linear Least-Squares method described in a next section, however this will only give us an approximation. A more systematic and accurate approach would take following steps:

1. Get initial estimates for $P$ with LLS
2. Correct projected image $\tilde{\boldsymbol{w}}$ by accounting for **radial** and **tangential distortions** [7]
3. Perform non-linear minimisation of an error term using Levenberg-Marquardt Algorithm [9], [10]

It is important to account for distortion, because it will improve the accuracy of localisation of calibration pattern in section Q1.3. **Radial distortion** arises from a lens shape, which is more rounded on the periphery than a parabola. **Tangential distortion** is due to manufacturing inaccuracy when aligning lens to be parallel to the CCD. This distortions can be accounted for by transforming $\tilde{\boldsymbol{w}} = [x_d\ y_d]$ to $\tilde{\boldsymbol{w}}_{\boldsymbol{p}} = [x_p\ y_p]$ as in Equation 17.

$$\begin{bmatrix} x_p \\ y_p \end{bmatrix} = \left(1 + k_1r^2 + k_2r^4 + k_3r^6\right) \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} 2p_1x_dy_d + p_2\left(r^2 + 2x_d^2\right) \\ p_1\left(r^2 + 2y_d^2\right) + 2p_2x_dy_d \end{bmatrix} \quad (17)$$

Having identified approximation for $\mathbf{P}$ and the distortion parameters $\mathbf{k} = [k_1\ k_2\ p_1\ p_2\ k_3]$ we can perform non-linear optimisation procedure (Equation 18) to minimise the error between projected image points and image points modeled with matrix $P$ and distortions $k$.

$$\min_P \sum_{i=1}^{m} \left\|\mathbf{w}_i - \check{\mathbf{w}}_i\left(\mathbf{P}, \mathbf{k}\right)\right\|^2 \quad (18)$$

## 2.3 Linear Least Squares

Matrix $P$ can be approximated by using linear least squares method on Equation 16. Presented in Equation 19 in a simplified form.

$$\mathbf{Ap} = 0 \quad (19)$$

LLS is performed by finding an eigenvector corresponding to the smallest eigenvalue of a square matrix $\mathbf{A}^T\mathbf{A}$. Such solution will correspond to a vector $\mathbf{p}$ which makes $\mathbf{Ap}$ closest to 0. It is important to note, that $\mathbf{p}$ needs to be constrained to unit length to avoid trivial solution $\mathbf{p} = 0$

### 2.4 Intrinsic and Extrinsic parameters

By switching from perspective to projective camera model we relaxed the constraint of matrix to take form $P = K[R|T]$. However it is still possible to covert back to that model. This can be done by applying **RQ-decomposition**, which outputs upper-triangular matrix $K$ (camera matrix) and matrix $R$ (rotation). Translation vector can be obtained via Equation 20.

$$\mathbf{T} = \mathbf{K}^{-1} \left(p_{14}, p_{24}, p_{34}\right)^T \tag{20}$$

Note, that camera matrix $K$ may have an extra $\gamma$ term in the second column of the first row. This parameter describes skewness between axes of the image plane, but for modern CCD cameras it usually equals 0.

## 3 Q 2.4

Expression for epipolar line can be derived from a relation between coordinate systems of the left camera $\mathbf{X'_c}$ and the right camera $\mathbf{X_c}$, both with orgins at corresponding optical centers. Figure 2 shows the camera setup.

$$\mathbf{X'_c} = \mathbf{R}\mathbf{X_c} + \mathbf{T} \tag{21}$$

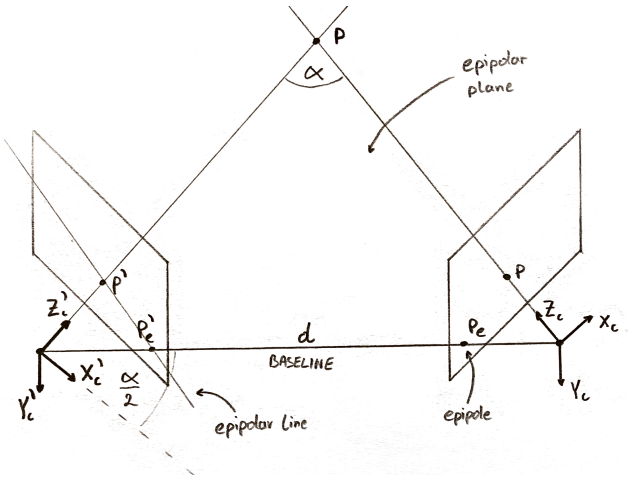



Figure 2: A schematic of stereoscopic pair of cameras (focal length f) arranged symmetrically with two optical axes coplanar and making angle $\alpha$

Equations 22 through 28 show can we can use relationship between coordinate systems to form an epipolar constraint. We start with applying vector product with $\mathbf{T}$ to both sides followed by dot product with $\mathbf{X'_c}$, to rearrange terms.

$$\begin{aligned} \mathbf{T} \times \mathbf{X'_c} &= \mathbf{T} \times \mathrm{R}\mathbf{X_c} + \mathbf{T} \times \mathbf{T} \\ \mathbf{T} \times \mathbf{X'_c} &= \mathbf{T} \times \mathrm{R}\mathbf{X_c} \end{aligned} \tag{22}$$

$$\begin{aligned} \mathbf{X'_c} \cdot (\mathbf{T} \times \mathbf{X'_c}) &= \mathbf{X'_c} \cdot (\mathbf{T} \times \mathrm{R}\mathbf{X_c}) \\ \mathbf{X'_c} \cdot (\mathbf{T} \times \mathrm{R}\mathbf{X_c}) &= 0 \end{aligned} \tag{23}$$

We can express vector product in an equivalent form

$$\mathbf{T} \times \mathbf{X}_\mathrm{c} = \mathrm{T}_\times \mathbf{X}_\mathrm{c} \tag{24}$$

$$\mathrm{T}_\times = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix} \tag{25}$$

We now define product $\mathrm{T}_\times \mathbf{X}_\mathrm{c}$ as a single matrix E which has a special name - Essential matrix.

$$\mathrm{E} = \mathrm{T}_\times \mathbf{X}_\mathrm{c} \tag{26}$$

$$\mathbf{X}_\mathrm{c}'^T \mathrm{E} \mathbf{X}_\mathrm{c} = 0 \tag{27}$$

Equation 27 also holds for rays $\mathbf{p}$ which are parallel to camera centered position vectors $\mathbf{X_c}$. Equation 28 is the epipolar constraint, it specifies that for a point $\mathbf{x}$ on the right image there is a point $\mathbf{x}'$ on the left image which must lie on a epipolar line defined by Equation 28.

$$\mathbf{p}'^T \mathrm{E} \mathbf{p} = \begin{bmatrix} x' & y' & f \end{bmatrix} \mathrm{E} \begin{bmatrix} x \\ y \\ f \end{bmatrix} = 0 \tag{28}$$

To compute E we need to identify transition vector T and rotation matrix R. Let's start with translation by the baseline $d$. Both origins of $\mathbf{X_c}$ and $\mathbf{X_c'}$ lie on a epipolar plane so there is not translation in $Y_c$. By noting that $X_c$ forms an angle $\alpha/2$ with the baseline we can write Translation matrix as in Equation 29 and also express $\mathrm{T}_\times$ in Equation 30.

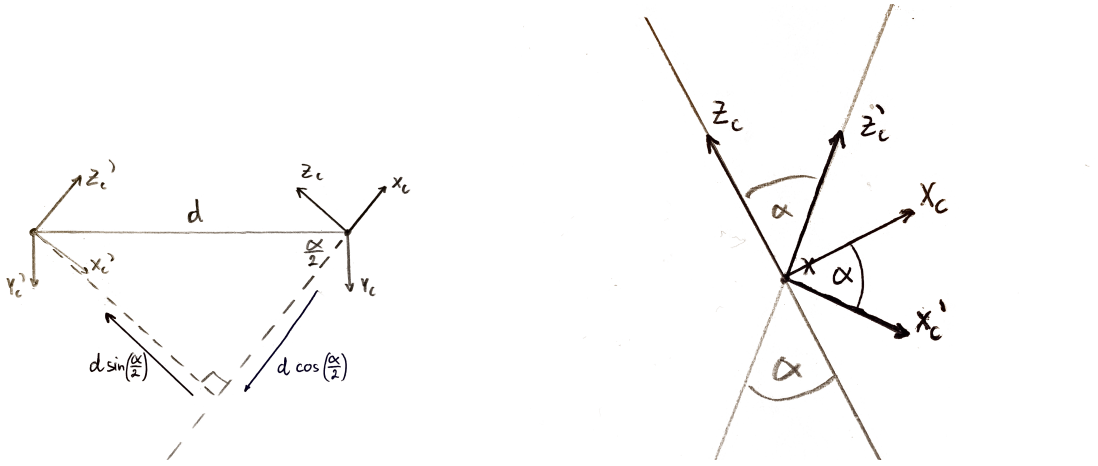



Figure 3: Schematics used to identify T and R

$$\mathbf{T} = \begin{bmatrix} d\cos(\frac{\alpha}{2}) \\ 0 \\ d\sin(\frac{\alpha}{2}) \end{bmatrix} \tag{29}$$

$$\mathrm{T}_\times = \begin{bmatrix} 0 & -d\sin(\frac{\alpha}{2}) & 0 \\ d\sin(\frac{\alpha}{2}) & 0 & -d\cos(\frac{\alpha}{2}) \\ 0 & d\cos(\frac{\alpha}{2}) & 0 \end{bmatrix} \tag{30}$$

Rotation matrix R can be identified by aligning origins and expressing $Z_c$ and $X_c$ in terms of the other coordinate system, as presented in Equation 31. There is no rotation along $\mathbf{X}_\mathrm{c}$ because $\mathbf{Y}'_\mathrm{c} = \mathbf{Y}_\mathrm{c}$ axis are parallel. Derived R is presented in Equation 32.

$$\begin{aligned} \mathbf{X}'_\mathrm{c} &= \mathbf{X}_\mathrm{c}\cos(\alpha) - \mathbf{Z}_\mathrm{c}\sin(\alpha) \\ \mathbf{Z}'_\mathrm{c} &= \mathbf{X}_\mathrm{c}\sin(\alpha) + \mathbf{Z}_\mathrm{c}\cos(\alpha) \end{aligned} \tag{31}$$

$$\mathrm{R} = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{32}$$

And the resultant Essential matrix in Equation 35 can be simplified using trigonometric identities

$$\begin{aligned} sin(A-B) &= sin(A)cos(B) - cos(A)sin(B) \\ cos(A-B) &= sin(A)sin(B) + cos(A)cos(B) \end{aligned} \tag{33}$$

$$\mathrm{E} = \mathrm{T}_\times\mathrm{R} = d\begin{bmatrix} 0 & -\sin(\frac{\alpha}{2}) & 0 \\ \cos(\alpha)\sin(\frac{\alpha}{2}) - \sin(\alpha)\cos(\frac{\alpha}{2}) & 0 & -\sin(\alpha)\sin(\frac{\alpha}{2}) - \cos(\alpha)\cos(\frac{\alpha}{2}) \\ 0 & \cos(\frac{\alpha}{2}) & 0 \end{bmatrix} \tag{34}$$

$$\mathrm{E} = d\begin{bmatrix} 0 & -\sin(\frac{\alpha}{2}) & 0 \\ -\sin(\frac{\alpha}{2}) & 0 & -\cos(\frac{\alpha}{2}) \\ 0 & \cos(\frac{\alpha}{2}) & 0 \end{bmatrix} \tag{35}$$

Having obtained Essential matrix we can finally write expression for epipolar lines on the left image $(x', y')$ in terms of $\alpha$, $f$, and image coordinates on the right image $(x, y)$.

$$\begin{bmatrix} x' & y' & f \end{bmatrix} \mathrm{E} \begin{bmatrix} x \\ y \\ f \end{bmatrix} = -y'\left(f\cos\left(\frac{\alpha}{2}\right) + x\sin\left(\frac{\alpha}{2}\right)\right) - x'y\sin\left(\frac{\alpha}{2}\right) + fy\cos\left(\frac{\alpha}{2}\right) = 0 \tag{36}$$

Simplifying and rearranging

$$y' = x'\left(\frac{-y\sin(\frac{\alpha}{2})}{f\cos(\frac{\alpha}{2}) + x\sin(\frac{\alpha}{2})}\right) + \left(\frac{fy\cos(\frac{\alpha}{2})}{f\cos(\frac{\alpha}{2}) + x\sin(\frac{\alpha}{2})}\right) \tag{37}$$

When $\alpha = 45°$

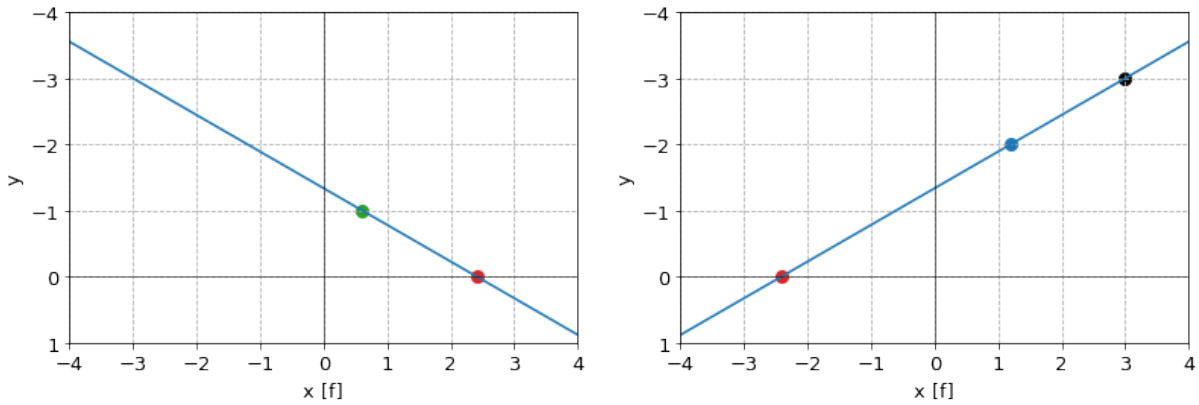
$$y' = x'\left(\frac{-y}{2.414f + x}\right) + \left(\frac{fy}{f + 0.414x}\right) \tag{38}$$

## 3.1 Pair of epipolar lines

Let's choose some arbitrary point in the 3D space $P$ which projects on to the right image at $\mathbf{p} = [3, -3, f]$. Equation 39 shows the corresponding epipolar line for that point on the left image. At $y' = 0$ the epipolar line passes through an epipole which corresponds to $\mathbf{p_e}' = [2.414f, 0, f]$.

$$y' = x'\left(\frac{3}{2.414f + 3}\right) + \left(\frac{-3f}{f + 0.414 \times 3}\right) \tag{39}$$

Figure 4.a shows the epipolar line. Now let's pick some arbitrary point $\mathbf{p}' = [0.610f, -1, f]$ lying on the epipolar line of the left image (Green dot) to identify a corresponding epipolar line on the right image. For simplicity we assume $f = 1$.




(a) Left camera (b) Right camera

Figure 4: Pair of epipolar lines. Where black point indicates initial point chosen on the right image. Red points indicate epipoles, and the Green point $[0.610f, -1, f]$ was used to plot epipole on the right image. Note: y-axis increases downwards

Figure 4.b shows the right epipolar line. The blue point and the epipole (red point) were used for plotting. The figure shows that right epipolar line contains a black point corresponding to $\mathbf{p} = [3f, 3, f]$. This proves that both epipolar lines lie on the same epipolar plane.

# 4 Q 2.5

## 4.1 SIFT keypoints detection

**Candidate keypoint localisation** SIFT uses local extrema of Difference of Gaussian (Equation 40) convolved with the image, as the initial candidates for keypoints (blob detection, see Q1.3).

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{40}$$

Maxima and minima are found by comparing $D(x, y, \sigma)$ value of each pixel with its 26 neighbouring pixels in the scale-space and selecting it as a keypoint only if it's the largest or the smallest. SIFT uses DoG as a function for features, due to low computational cost and high feature stability [8]. Stable features are desired because they are more likely to be detected in an image under different transformations. Preference for feature stability also guides empirical selection of sampling frequencies. SIFT samples 3 scales per octave and uses prior smoothing of $\sigma = 1.6$.

**Accurate keypoint localisation** Keypoint localisation can be improved by interpolating extrema with second order taylor expansion of $D(x, y, \sigma)$ (Equation 41). We evaluate it at a keypoint and shift it by $(x, y, \sigma)$ to the origin. The position of extremum $\hat{\mathbf{x}}$ of the resultant function indicates by how much we should adjust position of the keypoint, to improve its stability.

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^\mathbf{T}\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x} \tag{41}$$

Additionally, we use Equation 41 to identify keypoints with low contrast, which are sensitive to noise and thus should be rejected. These are the ones for which $|D(\hat{\mathbf{x}})| < tr$ where $tr$ is an empirically determined threshold (SIFT article uses $tr = 0.03$ for pixel values in range [0,1]).

We also want to reject keypoints along the edges, as these are hard to localise when noise is present. We identify them by taking ratio of $Trace(\mathbf{H})^2$ and $Det(\mathbf{H})$ of hessian matrix in Equation 42. We reject keypoints which don't satisfy inequality 43 as their ratio of the largest to smallest eigenvalue is greater than $r$, which is indicative of lying on an edge ($r = 10$ in SIFT article).

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{42}$$

$$\frac{\text{Trace}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} < \frac{(r+1)^2}{r} \tag{43}$$

## 4.2 SIFT keypoints descriptor

For features to be detectable under different image transformations, they have to be **scale (ii)** and **rotation invariant (i)**. First invariance is achieved by using Gaussian smoothed image with the closest scale to the scale of the keypoint, when constructing feature descriptors. Rotation invariance, is achieved by building feature descriptor relative to keypoint's orientation. Orientation is identified by constructing histogram of weighted intensity gradients of pixels in the region of the keypoint and using orientation of the dominant gradient as keypoint's orientation. If there are $n$ dominant gradients, then $n$ keypoint descriptors will be created for that keypoint.

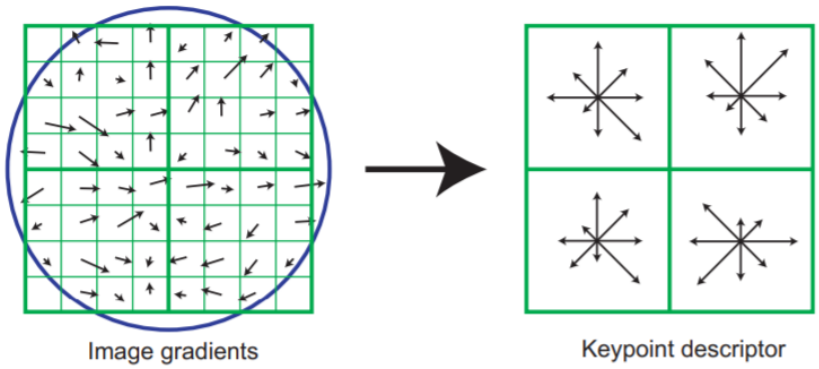



Figure 5: A schematic showing formation of 8x8 gradient matrix and 2x2 matrix of histograms. Note, that SIFT uses 16x16 matrix of gradients and 4x4 for histograms. Borrowed from SIFT article [Lowe, 2004]

Keypoint descriptors are created by taking a grid of gradients around the keypoint, weighting them with Gaussian function, and then subdividing into smaller regions where gradients are stacked to form a location invariant histograms (as presented schematically on Figure 5). Gradients are subdivided and stacked into histograms, which makes features invariant to spatial shifts of gradients within each sub-region. This is inspired by biological

vision and improves repeatability of detection from range of viewpoints **(iv)**. The Gaussian weighting is used to emphasise higher significance of gradients closer to the keypoint. This makes feature descriptors more robust to small spatial perturbations of the window **(iv)**. Empirically test showed that 16x16 grid of gradients and 4x4 matrix of histograms with 8 orientations give best results. The overall dimension of feature descriptor is $4 \times 4 \times 8 = 128$.

**Illumination invariance (iii)** is achieved by two-step process. First, we normalise 128 dimensional feature vector to achieve robustness to contrast changes (gradients are already invariant to brightness). Second, we threshold all feature values to 0.2 (found empirically) and renormalise. This provides invariance to non-linear illumination changes which can strongly influence gradient magnitudes, but not gradient orientations.

## 4.3 Alternative key point descriptor - GLOH

Gradient location and orientation histogram (GLOH) is an extension of SIFT algorithm aiming to increase SIFT's robustness and distinctiveness. GLOH and SIFT descriptor performance was compared by Mikolajczyk [11] and is summarised in a Table 1. Mikolajczyk evaluates descriptor performance for a range of region detectors. In Table 1, however, we considered only Hessian-Laplace and Hessian-Affine detectors, due to their similarity with the detector originally used in SIFT.

| | | type of image transformation | | | |
|---|---|---|---|---|---|
| | | SCALE | ROTATION | VIEWPOINT | ILLUMINATION |
| Structured scene | GLOH | ✓ | ✓ | ✓ | ✓ |
| | SIFT | | ✓ | | |
| Textured scene | GLOH | | ✓ | | N/A |
| | SIFT | ✓ | ✓ | ✓ | N/A |

Table 1: Comparison of performance of GLOH and SIFT descriptors under different image transformation conditions and scenes, using Hessian-Laplace (scale) and Hessian-Affine detectors (rotation, viewpoint, illumination)

We can compare two descriptors by looking at (*recall*) vs. $(1 - precision)$ plots for different image transformations. These plots shows what fraction of all feature pairs have we identified (recall) between original image and a transformed image, at a set precision level. Mikolajczyk study shows that GLOH, on average, has higher recall for structured scenes, while SIFT for textured scenes. This could be because GLOH considers more spatial regions for the histograms and thus can identify more distinct structure of regions. Yet, both GLOH and SIFT outperform non-region based feature descriptors when it comes to invariance to image transformations listed in Table 1.

# References

[1] Marr, D. and Hildreth, E. (1980). Theory of edge detection. Proceedings of the Royal Society of London. Series B. Biological Sciences, 207(1167), pp.187–217.

[2] H. Moravec (1980). "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover". Tech Report CMU-RI-TR-3 Carnegie-Mellon University, Robotics Institute.

[3] Harris, C. and Stephens, M. (1988). A Combined Corner and Edge Detector. Procedings of the Alvey Vision Conference 1988. [online] Available at: http://www.bmva.org/bmvc/1988/avc-88-023.pdf [Accessed 18 Nov. 2019].

[4] Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60(2), pp.91–110.

[5] Carr, P., Sheikh, Y. and Matthews, I. (2012). Point-less calibration: Camera parameters from gradient-based alignment to edge images. 2012 IEEE Workshop on the Applications of Computer Vision (WACV).

[6] Bradski, G. R., and Kaehler, A. (2008). Learning OpenCV. Beijing, O'Reilly.

[7] Brown, D. C. "Close-range camera calibration," Photogrammetric Engi- neering 37 (1971): 855–866.

[8] Mikolajczyk, K. 2002. Detection of local features invariant to affine transformations, Ph.D. thesis, Institut National Polytechnique de Grenoble, France

[9] Zhang, Z. "A flexible new technique for camera calibration," IEEE Trans- actions on Pattern Analysis and Machine Intelligence 22 (2000): 1330–1334.

[10] J. More. The levenberg-marquardt algorithm, implementation and theory. In G. A. Watson, editor, Numerical Analysis, Lecture Notes in Mathematics 630. Springer-Verlag, 1977.

[11] Krystian Mikolajczyk and Cordelia Schmid "A performance evaluation of local descriptors", IEEE Transactions on Pattern Analysis and Machine Intelligence, 10, 27, pp 1615–1630, 2005.