Assignment 3

Dana Kianfar - Jose Gallego Posada
11391014 - 11390689
{dana.kianfar, jose.gallegoposada}@student.uva.nl

1 Harris Corner Detection

1.1 Implementation

A good feature is defined by its *repeatability, saliency, compactness* and *locality*. Concretely, an ideal feature has information (gradients) in both x and y directions. The Harris Corner Detection (HCD) algorithm detects corners, which are generally good features by the aforementioned criteria.

This is achieved in a multi-stage process. The algorithm first computes the auto-correlation matrix of an image I. Let x and y denote the coordinates of a pixel in an image, and let u and v denote displacement in the x and y directions respectively. The auto-correlation function $E(u, v)$ for each pixel at (x, y) is defined as follows:

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

where $w(x, y)$ is a weighting function based on a neighborhood about (x, y). The auto-correlation function is approximated by its second order Taylor expansion centered around $u = 0, v = 0$:

$$E(u,v) \simeq E(0,0) + \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_u(0,0) \\ E_v(0,0) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{vu}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

where E_u, E_v denotes the first-order partial derivatives, and $E_{uu}, E_{uv}, E_{vu}, E_{vv}$ denote the second-order partial derivatives.

Note that at $u = 0, v = 0$, we can simplify the Taylor expansion as $E(0, 0) = E_u(0, 0) = E_v(0, 0) = 0$:

$$\begin{aligned} E(u,v) &\simeq \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} E_{uu}(0,0) & E_{uv}(0,0) \\ E_{vu}(0,0) & E_{vv}(0,0) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \\ &\simeq \frac{1}{2} \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} \sum_{x,y} 2w(x,y) I_x^2(x,y) & \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y) \\ \sum_{x,y} 2w(x,y) I_x(x,y) I_y(x,y) & \sum_{x,y} 2w(x,y) I_y^2(x,y) \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

$H = \sum_{x,y} w(x,y) \nabla I \nabla I^T$ is known as the second moment matrix. For H to be full rank it is necessary to have non-zero gradients in both x and y dimensions inside the neighborhood. Under this condition, two positive non-zero eigenvalues λ_1 and λ_2 can be found for H, and thus the pixel at (x, y) is a corner candidate. The quality of each candidate depends on the magnitudes of both eigenvalues. If either eigenvalue is small while the other is large, the candidate is more likely to be an edge than a corner. Only if both eigenvalues are large, the candidate is likely to be a corner.

HCD proposes a cornerness value defined by $R = \det(H) - \alpha \cdot \mathrm{Tr}^2(H)$, where $\alpha \in \mathbb{R}$ is a sensitivity parameter. The HCG algorithm then thresholds R to obtain candidate corner pixels. The threshold value can be defined by a heuristic:

$$\frac{c}{|X||Y|} \sum_{x \in X} \sum_{y \in Y} H(x, y)$$

where $c \in \mathbb{R}$ is a sensitivity parameter. HCD further filters the candidates using non-maximal supression to obtain the best candidate within each block of size $n \times n$ within the image.

1.2 Results

We present the results for our HCD implementation below. We use a Gaussian kernel as the weighting function $w(x, y)$ to obtain smoothed image gradients which are robust to texture and noise. Our implementation has the following six parameters:

- n: non-maximal supression window width
- T: threshold for corner map filtering (determined using aforementioned heuristic)
- c: scalar constant used in threshold heuristic
- σ: Gaussian smoothing kernel sigma parameter
- K: Gaussian kernel width
- α: HCD parameter for calculating cornerness

We experimented with different values of these parameters and display our results for two images *person_toy* and *pingpong* in Figures 1 and 3, respectively. In our visualizations, we also display MATLAB's `corners` function, and display separately the gradients for both images respectively in Figures 2 and 4.

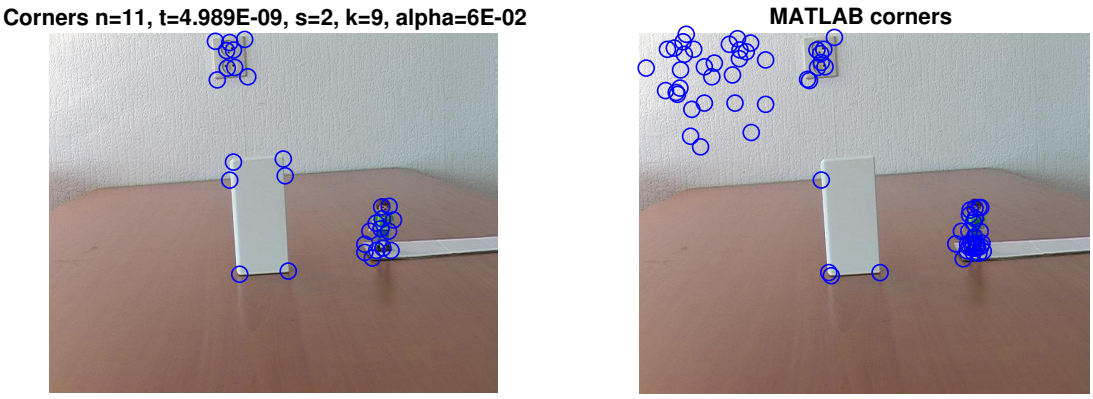


Figure 1: HCD results for person_toy

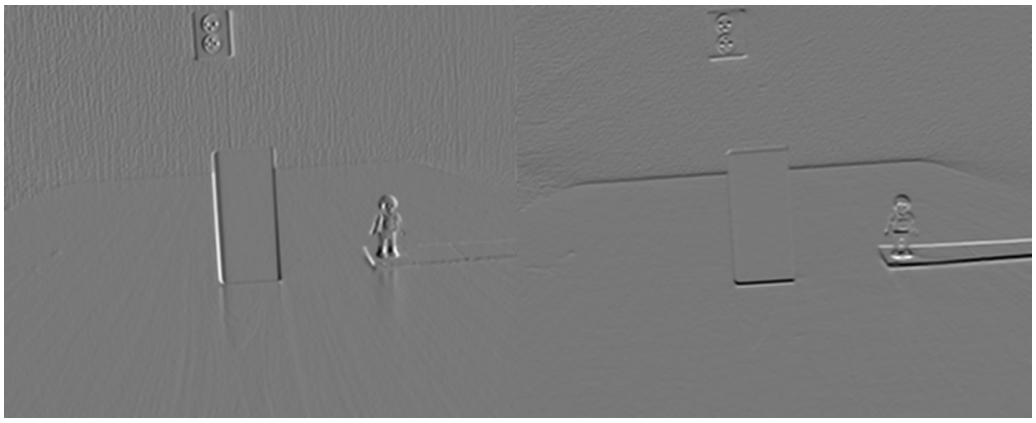


Figure 2: Image gradients in the x (left) and y (right) directions

We observe in Figure 1 that our implementation and parameter settings perform well. We note that the features we obtained are desirable as they 1. are few in number, 2. are mostly corners, 3. represent unique local regions within the image, 4. represent interesting parts of the image, and not general textures, such as the background wall. Our parameter settings are displayed on the figure.

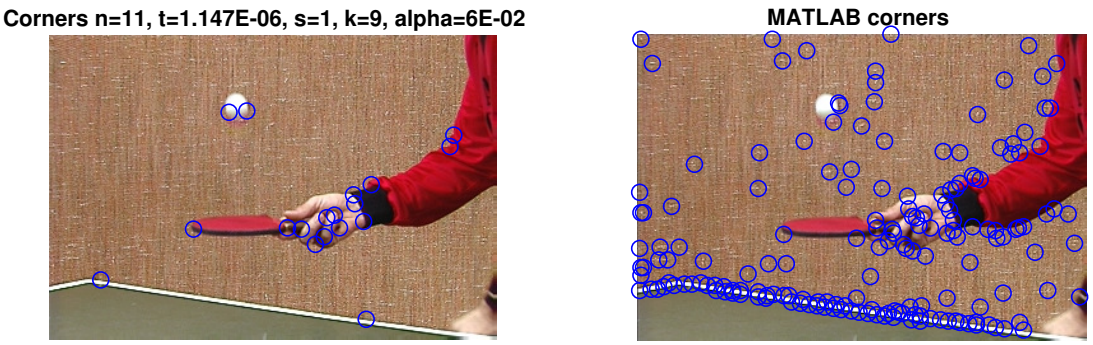


Figure 3: HCD results for pingpong

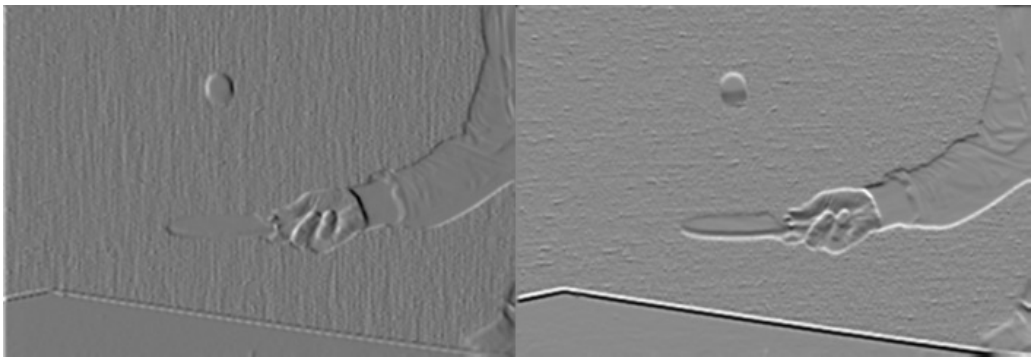


Figure 4: Image gradients in the x (left) and y (right) directions

Similarly, we observe in Figure 3 that our implementation and parameter settings are able to identify good features in the *pingpong* image.

1.3 Parameter Tuning

When experimenting with parameters, we were able to qualitatively observe the influence of each parameter on the quality of corner features. While parameter values can interact, we will attempt a rough summary of our observations in this section.

We observed that the number and quality of corner features is sensitive to smoothing, dictated by the width of the smoothing kernel K and sigma σ. The stronger the smoothing, the fewer corners we obtained, all other parameters kept constant. This was useful in smoothing out the texture of background walls in both images. With a smaller K we obtained too many features points as illustrated in Figure 5.

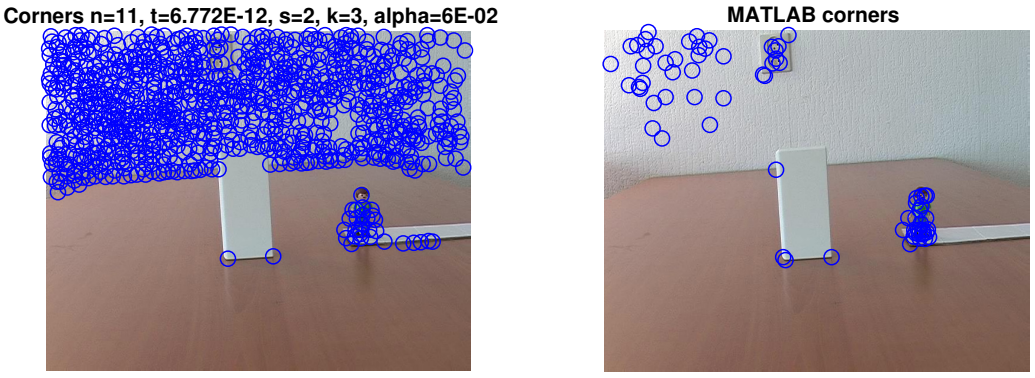


Figure 5: person_toy image with less smoothing

We also observed that a lower value of α can result in more feature points. As a sensitivity parameter, α is a scaling constant when evaluating the magnitudes of λ_1 and λ_2. We obtained more concise and high-quality features using $\alpha = 0.06$. The n parameter also affects how many corners are returned by the function, as it is the size of the neighborhoods

in the nonmaximal supression filtering. A very large n would return too few features as shown in Figure 6.

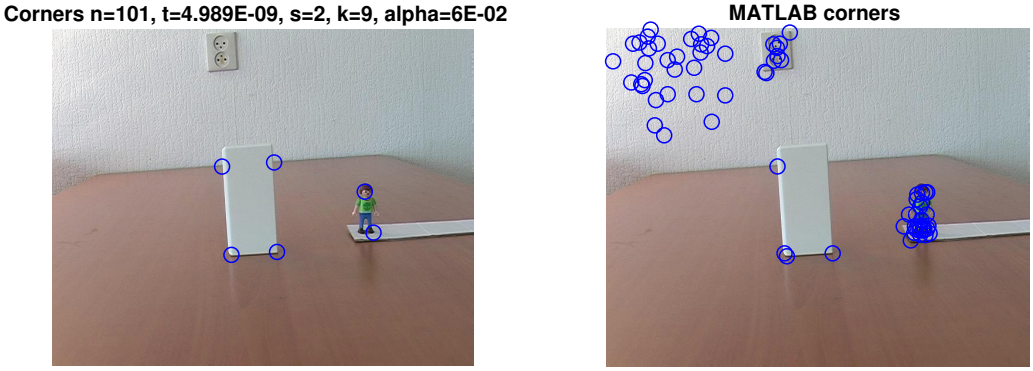


Figure 6: Toy image with a large n

We set $c = 1.5$ after experimenting with difference values. As expected, a smaller value of c will set a lower threshold for the corner map H values and therefore the algorithm will return more corners.

1.4 Shi and Tomasi

Shi and Tomasi proposed an alternative definition of the *cornerness* function $H(x, y)$ given by $\min\{\lambda_1, \lambda_2\}$, where λ_1 and λ_2 are the eigenvalues of $Q(x, y)$. Following the previous notation, it is clear that:

$$Tr(Q) = A + C = \lambda_1 + \lambda_2$$

$$|Q| = AC - B^2 = \lambda_1 \lambda_2$$

We claim that the min function can be written as:

$$\min\{\alpha, \beta\} = \frac{(\alpha + \beta) - \sqrt{(\alpha - \beta)^2}}{2}$$

Without loss of generality, let $\lambda_1 = \lambda_2 + r$ for some $r \geq 0$. The previous relation holds, since:

$$\min\{\lambda_1, \lambda_2\} = \frac{(\lambda_1 + \lambda_2) - \sqrt{(\lambda_1 - \lambda_2)^2}}{2} = \frac{(\lambda_2 + r + \lambda_2) - \sqrt{(\lambda_2 + r - \lambda_2)^2}}{2} = \lambda_2$$

Now, we can express this in terms of A, B and C, without performing eigenvalue decompositions:

$$\begin{aligned} \min\{\lambda_1, \lambda_2\} &= \frac{(\lambda_1 + \lambda_2) - \sqrt{(\lambda_1 - \lambda_2)^2}}{2} \\ &= \frac{(\lambda_1 + \lambda_2) - \sqrt{\lambda_1^2 - 2\lambda_1\lambda_2 + \lambda_2^2}}{2} \end{aligned}$$

$$
\begin{aligned}
&= \frac{(\lambda_1 + \lambda_2) - \sqrt{(\lambda_1 + \lambda_2)^2 - 4\lambda_1\lambda_2}}{2} \\
&= \frac{(A + C) - \sqrt{(A + C)^2 - 4(AC - B^2)}}{2} \\
&= \frac{(A + C) - \sqrt{A^2 + 2AC + C^2 - 4AC + 4B^2}}{2} \\
&= \frac{(A + C) - \sqrt{(A - C)^2 + 4B^2}}{2}
\end{aligned}
$$

Shi and Tomasi consider the $\min\{\lambda_1, \lambda_2\}$ (see Figure 7), therefore the cornerness value is low if at least one of the eigenvalues is smaller than λ_{min} (orange and gray regions), and large if both eigenvalues are larger than λ_{min} (green region).

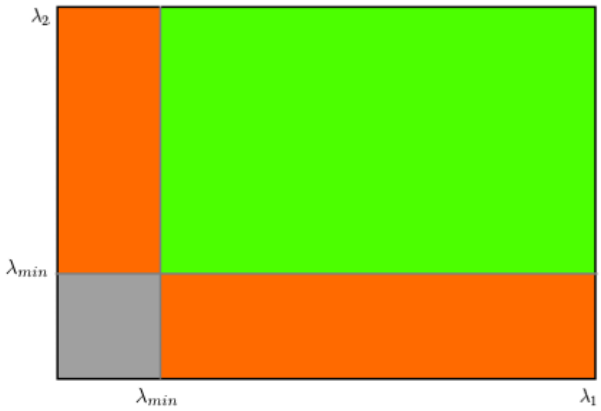


Figure 7: Shi-Tomasi cornerness visualization. Large cornerness values are present in the green region. Taken from [1].

2 Lucas-Kanade Algorithm

2.1 Implementation

Let $I(x, y, t)$ be the intensity at the image on pixel (x, y) at time t. In the basic Optical Flow model, each pixel is assumed to satisfy the *brightness constancy constraint*:

$$\nabla I \cdot \begin{bmatrix} V_x \\ V_y \end{bmatrix} = I_x V_x + I_y V_y = -I_t$$

where V_x and V_y are the x and y components of the optical flow of $I(x, y, t)$, and I_w represents the derivative of the image in the w direction.

Note that the previous is an underconstrained problem for each pixel since we have one equation and two variables, V_x and V_y. The Lucas-Kanade (LK) algorithm assumes

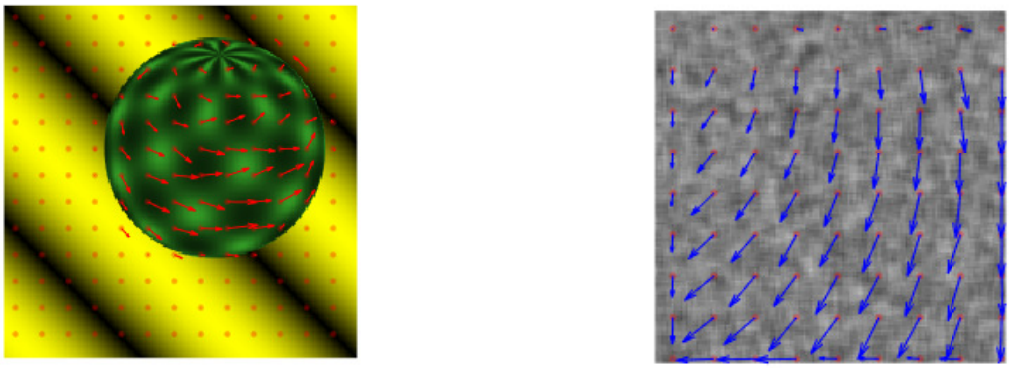
that the optical flow is constant in a vicinity of the pixel under consideration, and thus forms the following system:

$$\begin{bmatrix} \nabla I(p_1)^T \\ \vdots \\ \nabla I(p_n)^T \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ \vdots \\ I_t(p_n) \end{bmatrix}$$

where $\{p_i\}_{i=n}^{n}$ is the set of points in the neighborhood of (x, y). Finally, this system is solved using the usual least-squares pseudo-inverse technique.

2.2 Results

In our experiments, we use non-overlapping regions in the image of size N, and perform smoothing with a Gaussian filter of width K and amplitude σ before calculating the derivatives of the images. The results of this technique are displayed in Figure 8.



(a) Sphere

(b) Synth

Figure 8: LK optical flow results with $N = 15$, $K = 9$ and $\sigma = 1$.

The red points represent the center of each non-overlapping region and the arrows show the optical flow vector at each point. This technique successfully detects motion in the pixel between the two provided frames on both images.

We note that with in Figure 10a with a region size of $N = 15$, there is noise in the orientation of the arrows. We attribute this to the aperture problem in flat or edge-like regions. This effect can be see more clearly in Figure 9 where we increased the granularity of the grid. The arrows at the center of the green and black patches are noisy and point in varying directions, whereas the arrows on the corners of the patches, i.e. where a gradient exists, are mostly consistent in direction and magnitude.

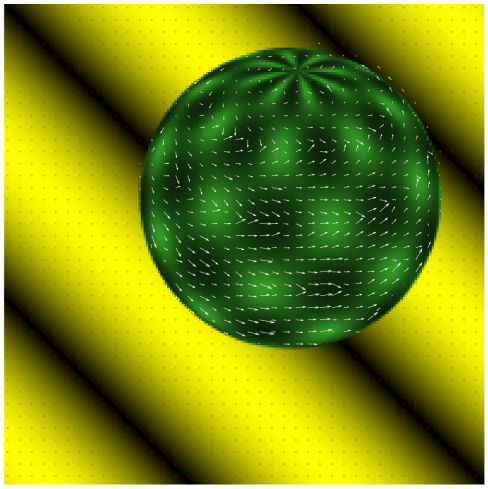


Figure 9: LK optical flow results with $N = 5$, $K = 9$ and $\sigma = 1$.

2.3 Horn-Schunk Smooth Optical Flow

A variant of optical flow was proposed by Horn and Schunk [2]. It is known that Lucas-Kanade fails in flat regions where the derivatives are ambiguous (close to zero) due to the aperture problem. Such instances result in matrix A becoming singular. To counter this problem, the Horn-Schunk (HS) method makes an additional assumption that motion is smooth across the image, i.e. their derivatives are very close to zero. This is expressed as global constraint across the entire image, formulated directly as an error minimization problem expressed below

$$E = \int_x \int_y [(I_x u + I_y v + I_t)^2 + \alpha^2(||\nabla_u||^2 + ||\nabla_v||^2)] dy\, dx$$

where E is an error function which estimates the violation of the brightness constancy and smooth motion constraints. As E is measured over the entire image, it is a global constraint.

The HS method produces smooth flow vectors in flat regions where the local gradients are zero. This is achieved by using global information, and is the reason why the HS method is robust to the aperture problem [3].

3 Tracking

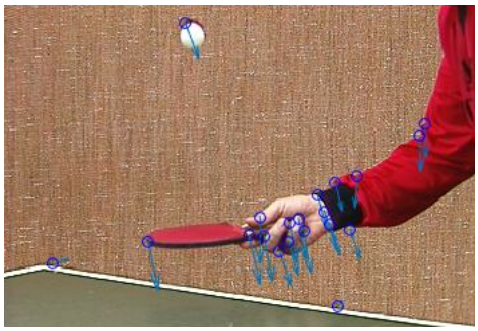
3.1 Implementation

In this section we implement a simple tracking algorithm using the results from previous sections. Every T frames we detect features using HCD and on every frame transition

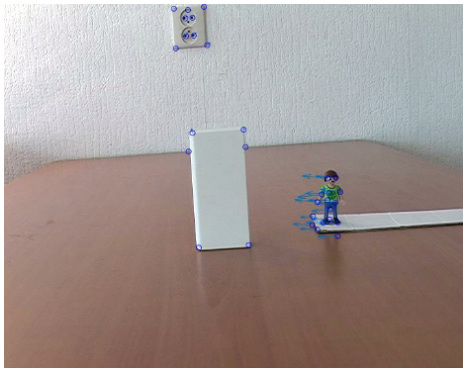
we predict the position of each detected feature using the estimated optical flow with a neighborhood of size N around each feature. The rationale for this approach is two-fold: first, across frames some of the features might get occluded, which complicates the tracking task; second, the repeatability characteristic of the HCD ensures that we will find roughly the same features on nearby frames.

3.2 Results

The **.avi* for each frame sequence can be found in the videos folder under the corresponding name. Sample snapshots of the resulting videos are displayed in Figure 10.



(a) Ping-pong



(b) Toy

Figure 10: Tracking (HDC + LK) results with $N = 25$. HDC parameters and LK smoothing per image as in Section 1.

Initially we performed feature detection only in the first frame. However, occlusions or non-smooth changes between frames induce a high error rate in the prediction of the feature position after motion, as can be seen with the fast movement of the ping pong ball. We also observed that the scaling factor applied to the optical flow vectors has a critical impact in the performance of the system. However tuning this parameter is fairly complicated and a better performance was achieved by detecting features periodically.

References

[1] OpenCV, "Shi-Tomasi Corner Detector & Good Features to Track - OpenCV Documentation," 2017. [Online]. Available: http://docs.opencv.org/3.0-beta/_images/shitomasi_space.png

[2] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185 – 203, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0004370281900242

[3] R. Radke, *Computer Vision for Visual Effects*, ser. Computer Vision for Visual Effects. Cambridge University Press, 2013. [Online]. Available: https://books.google.nl/books?id=MgIVyAWf9VUC