David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

1. Optical Flow

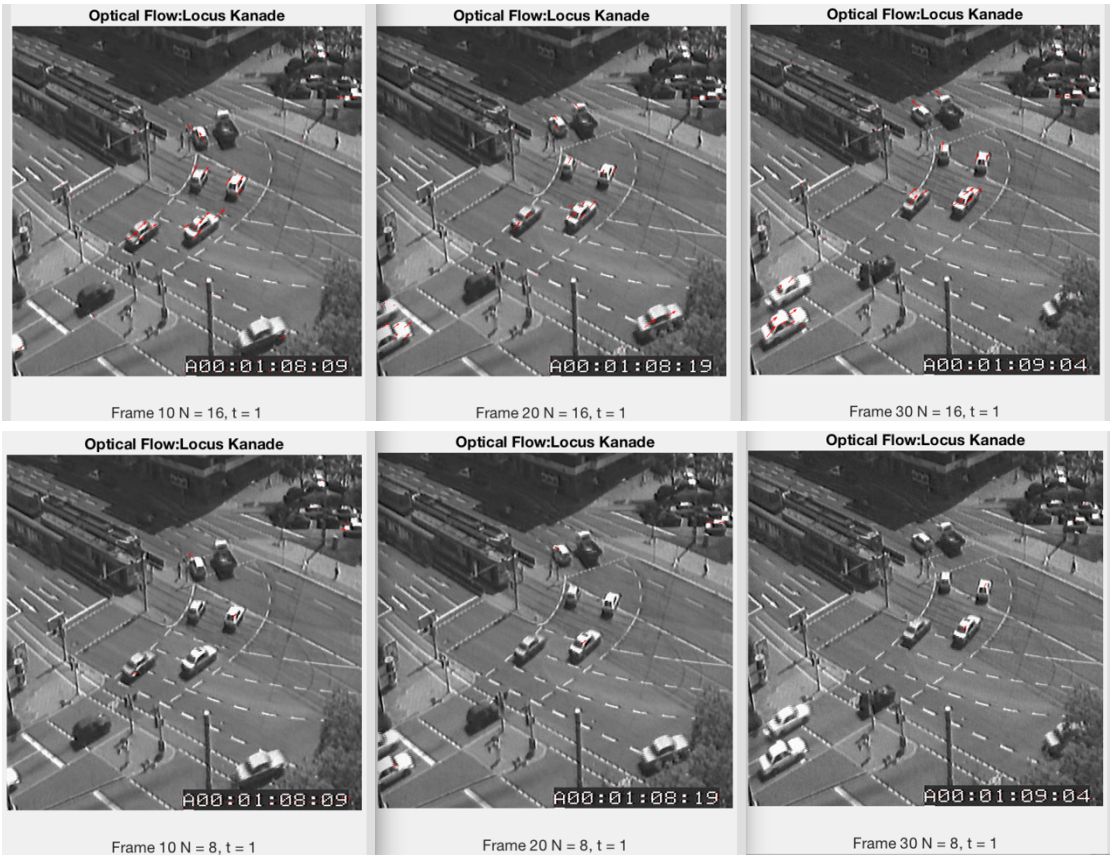
The process involves inverting $A^T A$. Note that an ill-conditioned $A^T A$ (small eigenvalues), caused by the summation of a small image-region (or a larger region with small image intensity-gradients), will lead to numerical errors and sensitivity to noise.

A large window N allows to capture the entire object and extract the optical flow more reliably. However, a large window N may cover several objects moving in different directions, i.e. multiple motions within a finite region (leading to less accurate flow).

A larger window N would also provide better robustness to noise, since more gradients are summed towards larger values of $A^T A$ - leading to higher probability of larger eigenvalues that meet the threshold and ensure numeric stability; better separation of the signal from the noise.

A smaller window size N may capture only edges ($A^T A$ becomes singular) or that the window covers a homogeneous portion of the image (small eigenvalues of $A^T A$).

The figure below demonstrates the effect of a large window (upper-row) vs a small window (bottom row).

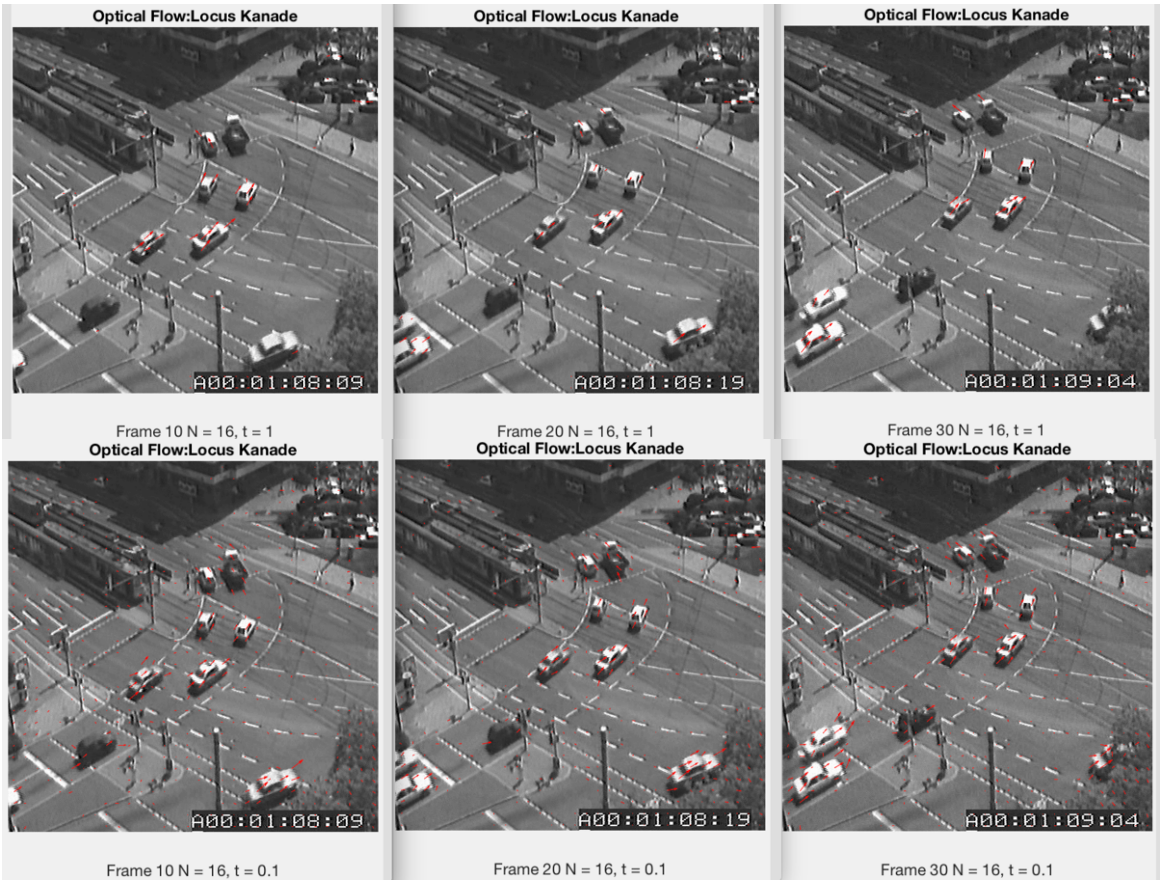


Optical flow at frames 10,20,30
window size N = 16 (top) vs N = 8 (bottom)
for threshold t = 1.
The larger window “captures” more of the optical flow

David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

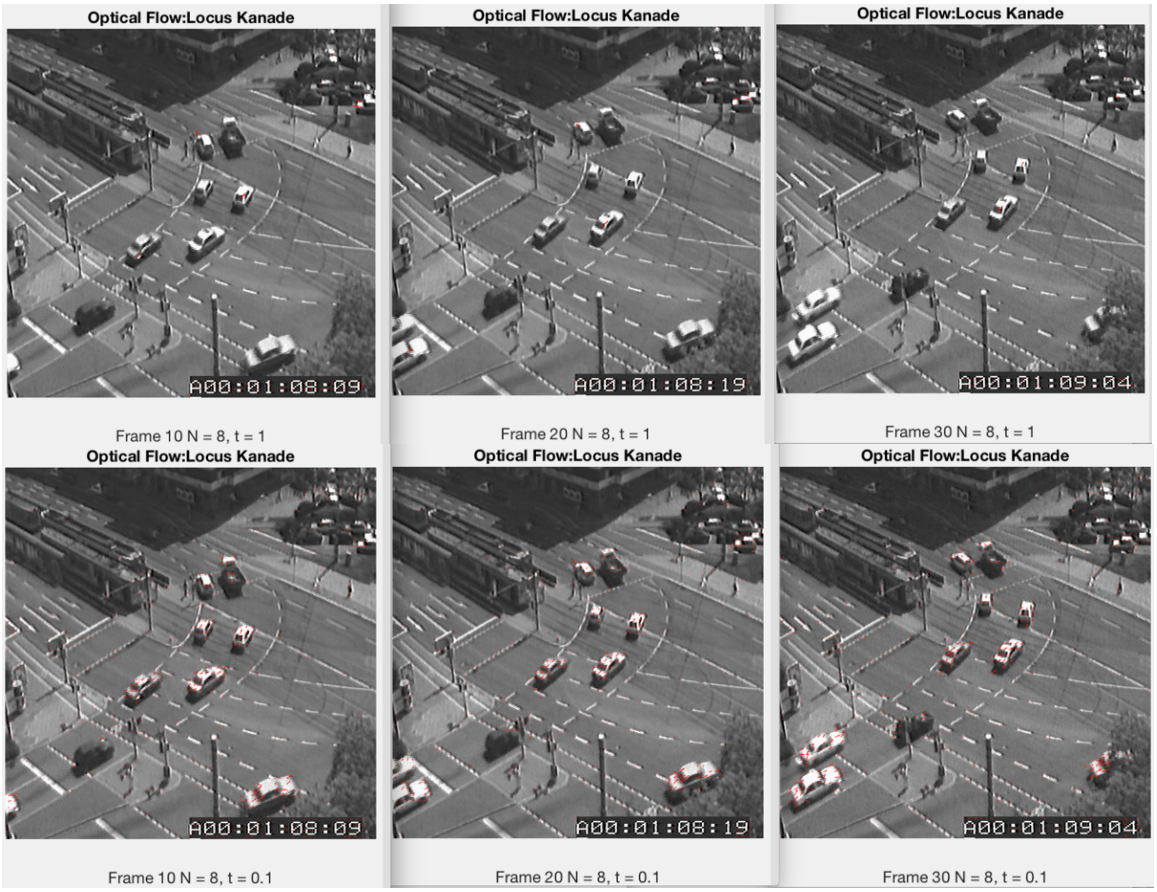
Decreasing the threshold t leads to a more detailed optical flow (more arrows) at the cost of susceptibility to image noise and may suffer from numerical instability.
A higher threshold t may discard valuable flow information.

The figure below demonstrates the effect of a high threshold (upper-row) vs low threshold (bottom row).



Optical flow at frames 10,20,30 window size N = 16 for threshold t = 1 (top) vs t = 0.1 (bottom)

David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il



Optical flow at frames 10,20,30 window size N = 8 for threshold t = 1 (top) vs t = 0.1 (bottom)

As can be seen from the images, selecting a suitable window-size and threshold is likely to produce substantially better results: If the window is too small, it will not reliably capture the motion in the frame – and if the threshold is too high, it will discard much of the valuable flow information. Setting the threshold too low will introduce noise and numerical instability, distorting the results.

David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

2. Camera Calibration

2.a.

The obtained projection matrix $P = K[R|t] = K[R|-RC] = [M|-MC]$ using DLT:

$$P = \begin{bmatrix} 0.0057 & 0.0826 & -0.0010 & -0.8674 \\ 0.0101 & -0.0001 & 0.0823 & -0.4907 \\ -0.00008 & 0.00001 & 0.000001 & -0.0006 \end{bmatrix}$$

2.b.

The world points were re-projected on to the image plane using the estimated projection matrix $\tilde{x} = PX$.
Both sets of 2d points were normalized based on the w component.

We used the mean Euclidian distance measured between $\tilde{x}$ and the reference 2d points in the image plane x as an error measure:
An estimation error of 3.522932 was obtained. The error has no units since the points are homogeneous.
The figure below shows the points of x (blue) and $\tilde{x}$ (red) overlaid on the image (left) and a zoom-in view highlighting the estimation error (right).

David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

2.c.

Our goal is to decompose M into KR where K is the calibration matrix (intrinsic camera parameters) and R is the unitary rotation matrix.
To this end, we decompose $M = RQ$ where R is an upper-triangular matrix and Q is a unitary orthogonal matrix. In addition, K has positive values in its diagonal.
The RQ decomposition is Gram-Schmidt orthogonalization of rows of M, starting from the last column – Resulting in an upper-triangular matrix times the unitary matrix.

Since the RQ decomposition isn't available in Matlab, we opt to use Matlab's QR decomposition, which results in a unitary matrix times an upper-triangular matrix.

Attempting to perform the QR decomposition on M^T would yield $R^T Q^T$ where R^T is a lower-triangular matrix, whereas the calibration matrix K should be upper triangular. In order to obtain RQ from the QR decomposition, we'll employ matrix transposition and permutation: The permutation matrix P reverses the rows of M when multiplied from the left (flip upside-down) and reverses the columns of M when multiplied from the right (flip left-to-right).

$$P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \qquad P = P^T = P^{-1}, \qquad PP^T = I_n$$

The equality $M = P^T((PM)^T)^T$ allows us to perform QR decomposition of $(PM)^T$ to obtain QR, followed by a transpose and multiplication by P^T on the left to restore M.

Starting with $(PM)^T$ – the transpose of the reversed rows of M, the QR decomposition produces $(QR)^T = R^T Q^T$.

```
[Q,R] = qr(flipud(M)');
```

We then transform $\hat{R} = P^T R^T P^T$ meaning reversing the order of the rows of R^T (upside down) followed by the reversal of its columns (left -to-right):
$\hat{R}$ is upper-triangular.

```
R = flipud(R');
R = fliplr(R);
```

For the unitary matrix Q, we perform $\hat{Q} = PQ^T$. The matrix $\hat{Q}$ is unitary orthogonal.

```
Q = Q';
Q = flipud(Q);
```

Finally, we ensure $\hat{R}$ has a positive diagonal by multiplying it by $T = diag(sign(diag(R)))$ and factorizing $\hat{Q}$ by the same matrix T

```
T = diag(sign(diag(R)));
R = R * T;
Q = T * Q;
```

To illustrate:
$M = P^T((PM)^T)^T = P^T(QR)^T = P^T R^T Q^T = P^T R^T P^T P Q^T = \hat{R}\hat{Q} = \hat{R}TT\hat{Q} = \tilde{R}\tilde{Q}$

David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

2.d.

The recovered calibration matrix K has reasonable parameters after it is normalized by $K(3,3)$ to ensure it takes a value of 1:

$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 10{,}005 & 70 & 1{,}224 \\ 0 & 10{,}032 & -1{,}053 \\ 0 & 0 & 1 \end{bmatrix}$$

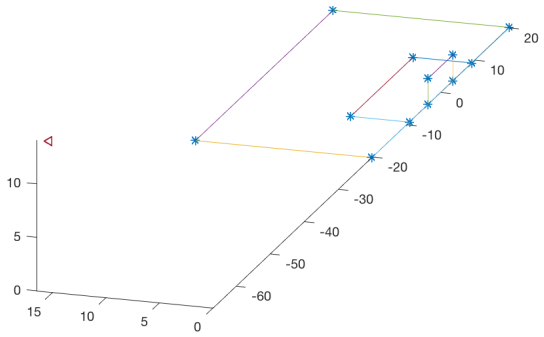
- The matrix K has positive diagonal values and $K(3,3) = 1$.
- The per-axis focal lengths maintain $f_x \approx f_y$ (meaning near symmetric pixels).
- The skew is negligible (by two orders of magnitude) compared to the other parameters of the calibration matrix, indicating almost-orthogonal axis of the sensor.
- The optical center is at $(c_x, c_y) = (1{,}224, -1{,}053)$

2.e.

The obtained rotation matrix R has $det(R) = -1$ and so requires negating the matrix to ensure that $det(R) = 1$. Beyond that, the values are reasonable as R is orthonormal: R maintains $R^T R = I$

2.f + g.

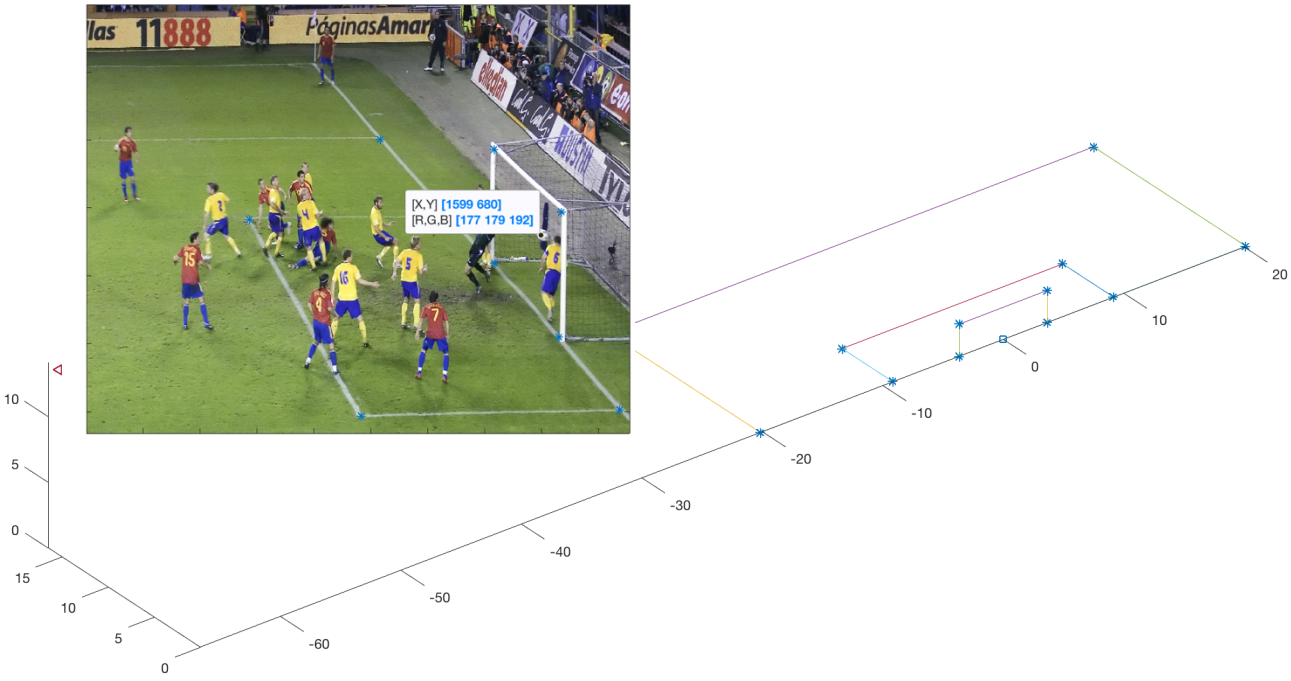
The recovered translation vector in the world space $c = [-66.6598\ 15.3006\ 14.1360]$ indicating that the camera is positioned above the XY plane (positive Z value) on the near side of the field (negative X value) and before the goal line (positive Y value).
This matches the view-point in the image, as shown below (left). The figure below also illustrates the $3D$ field model (right), with the camera's position marked as ◁



David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

2.h.

Our attempted to determine the 3D location of the ball involved visually annotating the center of the ball in the image and re-projecting the 2D point onto the 3D model.
From the supplied image it would appear that the center of the ball was situated directly above the goal line (marked below as ■), meaning that a portion of the ball may have crossed the goal line.
Our conclusion is that it is difficult to determine the exact position of the ball:
Rays originating from the optical center of the camera in the direction of the ball, would pass through the ball and cross into the goal-post. We cannot discern where along these rays the ball is situated (the depth of the image).
This implies that we need additional information. Perhaps another viewing angle of the same scene, or even knowledge of the dimensions of the ball and other real-world elements (such as the goal-post, the adjacent players etc.).



David Weinstein 302470596 dudiwa@campus.technion.ac.il
Yahel Kleinman 308010743 yahelk@campus.technion.ac.il

Depth with stereo

Disparity Map and Stereo Depth Estimation. The Disparity map shows the differences of each pixel, between the two (translated) stereo images. The disparity has a reverse ratio with the depth ($Z - axis$), so ultimately, this map is also a map of Depth.
Rectified images are easier to calculate their scan lines of the candidate corresponding pixels because those lines are on the Epipolar Lines, which in the case of the rectified images, are on the $x - axis$ direction and parallel on the $y - axis$.
Constructing the disparity map: We assume that for the rectified images, the pixel on the left image will be with higher values (direction right on the $x - axis$) than the corresponding pixel on the right image ($disparity > 0$).
Moreover, we assume that the distance between the two pixels won't be higher than 60 pixels ($disparity_{max} \leq 60$).
Experiments: We evaluate our algorithm on 3 pictures and calculate the error between the ground truth and our disparity map (RMSE). We can see that in most cases, the mid-sized window (11 pixels) achieves the best balance of precision and noise reduction; it often yields the best results in terms of RMSE error.