mathematics

MDPI

Article

Identification of Location and Camera Parameters for Public Live Streaming Web Cameras

Aleksander Zatserkovnyy [1,*] and Evgeni Nurminski [2]

[1] Pacific Oceanological Institute of Far Eastern Branch of RAS, 690041 Vladivostok, Russia
[2] Center for Research and Education in Mathematics, Institute of Mathematics and Computer Technologies, Far Eastern Federal University, 690922 Vladivostok, Russia
* Correspondence: avz@poi.dvo.ru

Abstract: Public live streaming web cameras are quite common now and widely used by drivers for qualitative analysis of traffic conditions. At the same time, they can be a valuable source of quantitative information on transport flows and speed for the development of urban traffic models. However, to obtain reliable data from raw video streams, it is necessary to preprocess them, considering the camera location and parameters without direct access to the camera. Here we suggest a procedure for estimating camera parameters, which allows us to determine pixel coordinates for a point cloud in the camera's view field and transform them into metric data. They are used with advanced moving object detection and tracking for measurements.

Keywords: depth map; radial distortion; perspective distortion; camera calibration; transport traffic statistics

MSC: 37M10; 65D18; 68U10; 90B20

Citation: Zatserkovnyy, A.; Nurminski, E. Identification of Location and Camera Parameters for Public Live Streaming Web Cameras. *Mathematics* **2022**, *10*, 3601. https://doi.org/10.3390/math10193601

Academic Editor: José Antonio Sanz

Received: 5 August 2022
Accepted: 23 September 2022
Published: 1 October 2022

1. Introduction

There are many ways to measure traffic [1–3]. The most common in the practice of road services are radars combined with video cameras and other sensors. The measuring complexes are above or at the edge of the road. Inductive sensors are the least dependent on weather conditions and lighting. The listed surveillance tools assume installation by the road or on the road. Providers of mobile navigator applications receive data about the car's movement from the sensors of mobile devices. Autonomous cars collect environmental information using various sensors, including multiview cameras, lidars, and radars. The results of the data collection systems of road services, operators of navigators, and autonomous cars are usually not available to third-party researchers. Notably, pioneering traffic analysis work describes processing video data recorded on film [3] (pp. 3–10).

Operators worldwide install public web cameras, many of which look at city highways; e.g., there are more than a hundred similar cameras available in Vladivostok.

A transport model verification requires actual and accurate data on transport traffic, covering a wide range of time intervals with substantial transport activity. Public live-streaming cameras can be a good and easily accessible source of data for that kind of research. Of course, this accessibility is relative. Video processing involves storing and processing large amounts of data.

The ref. [4] demonstrates road traffic statistics collection from a public camera video, where the camera has little perspective and radial distortions in the region of interest in the road (Region Of Interest, ROI). However, the distortions make significant changes in images for the majority of public cameras. The current article generalizes this approach to the case where a camera has essential radial and perspective distortions.

Street camera owners usually do not announce camera parameters (focal length, radial distortion coefficients, camera position, orientation). Standard calibration procedures with

a pattern rotation can be useless for cameras on a wall or tower. In this case, we suggest the implementable camera calibration procedure, which uses only online data (global coordinates of some visible points, photos, and street view images).

With known camera parameters, we construct the mapping between the ROI pixel coordinates and metric coordinates of points on the driving surface, which gives a way to estimate traffic flow parameters in standard units such as car/meter for traffic density and meter/second for car velocity with improved accuracy.

2. Coordinate Systems, Models

We select an ENU coordinate frame F_{enu} (East, North, Up) with an origin on a fixed object to localize points $P = (x_e, y_e, z_e)$, where x_e, y_e, and z_e are coordinates of P in F_{enu}.

A camera forms an image in the sensor (Figures 1 and 2). A digital image is a pixel matrix $N \times M$ (N columns, M rows). A pixel position is described in the image coordinate frame F_{im} in the image plane $P_{im} = (u, v)$ where u and v are real numbers, coordinates of P_{im} in F_{im}. Axis U corresponds to the image matrix rows (towards the right), and axis V corresponds to columns (from up to down). The camera orientation determines the image sensor plane orientation. Integer indexes define pixel position in the image matrix. We can obtain them by rounding u and v to the nearest integers. Let w and h be the image pixel width and height, respectively. For the image width and height, we have $W = Nw$ and $H = Mh$.

Camera position and orientation define the camera coordinate frame F_c (Figure 1). The camera perspective projection center O is the origin of F_c. We denote by (O_{xe}, O_{ye}, O_{ze}) the O coordinates in F_{enu}. Axis Z_c follows along the camera optical axis, axis X_c is parallel to axis U, and Y_c is parallel to axis V. Axis X_c defines the camera horizon. The deviation of X_c from the horizontal plane $X_{enu}Y_{enu}$ defines the camera horizon tilt. Unlike F_{im}, the F_{enu} and F_c are metric coordinate frames. F_{im} coordinates (u, v) approximate matrix indexes. To obtain meters from u on axis U, we use the expression $m(u) = wu$, and to obtain meters from v on axis V, we use $m(v) = hv$.

Suppose the camera builds an image close to a perspective projection of a 3D scene. We describe this camera transformation as the pinhole camera model (Figures 1 and 2 and ref. [5]). A point image $P_{im} = (u, v)$ is the transformation of the original 3D point P coordinates described in F_{enu}. The matrices A and $[R|t]$ define this transformation:

$$A = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

$$[R|t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}. \tag{2}$$

The triangular matrix A contains the intrinsic parameters of the camera. Camera position and orientation determine matrix $[R|t]$. Matrix $[R|t]$ defines transformation from frame F_{enu} to camera coordinates F_c:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = [R|t] \begin{bmatrix} x_e \\ y_e \\ z_e \\ 1 \end{bmatrix}, \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = -R \begin{bmatrix} O_{xe} \\ O_{ye} \\ o_{ze} \end{bmatrix}. \tag{3}$$

Here, R is a rotation matrix, and t is the shift vector from the origin of F_{enu} to O (origin of F_c). Matrix A includes the principal point C coordinates (c_u, c_v) in frame F_{im} and the camera focal length f divided by pixel width and height:

$$f_u = f/w, \; f_v = f/h. \tag{4}$$

The camera's optical axis and the image plane intersect at C. It is the spatial point G image (Figures 1 and 2). Usually, the coordinates (c_u, c_v) point to the image sensor matrix center (e.g., full HD resolution is 1920×1080, so $c_u = 960, c_v = 540$). Some modes of cameras can produce cropped images shifted from the principal point; we do not consider this case here.

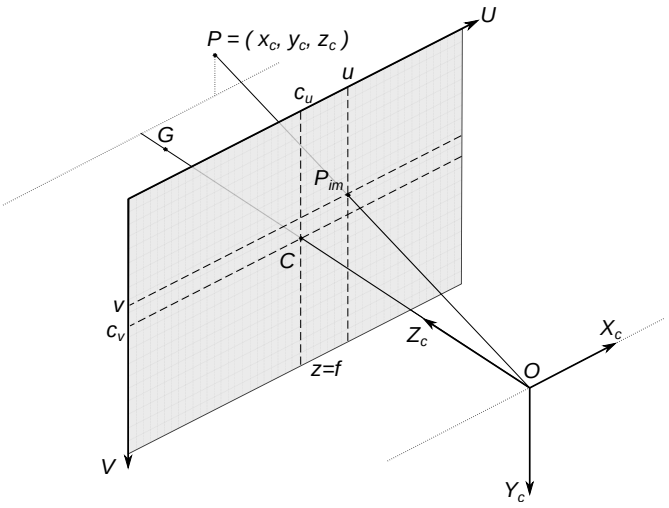

Figure 1. Coordinate frames F_c, F_{im}. P_{im} are the image of the 3D point P. Axis Z_c follows the camera optical axis. The camera forms a real image on the image sensor behind aperture O in the plane $z = -f$; however, the equivalent virtual image in the plane $z = f$ is preferable in illustrations.

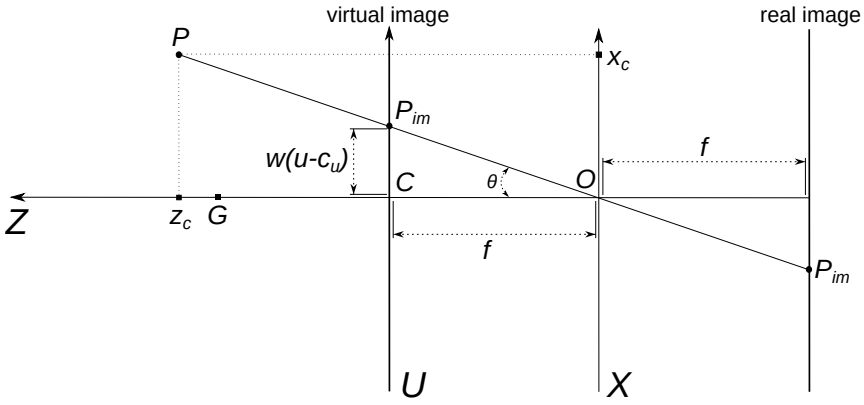

Figure 2. Virtual and real images for a camera with aperture O. The scene is projected on the plane XZ of frame F_c. The coordinates of the point P in F_c are denoted by x_c, z_c, and the u, c_u are coordinates of points P, C on the axis U of frame F_{im}.

The triangles (O, P_{im}, C) and $(O, P, (z_c, 0, 0))$ are similar on the plane ZX_c and on ZY_c (Figure 2), so

$$\begin{array}{l} x_c/z_c = w(u - c_u)/f = \tan\theta_x \\ y_c/z_c = h(v - c_v)/f = \tan\theta_y \end{array}. \tag{5}$$

It follows from (5) that pixel coordinates in F_{im} are connected with coordinates of the spatial source in F_c by the equations

$$P_{im}^h = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} x_c/z_c \\ y_c/z_c \\ 1 \end{bmatrix} = \begin{bmatrix} f_u x_c/z_c + c_u \\ f_v y_c/z_c + c_v \\ 1 \end{bmatrix}, \tag{6}$$

where P_{im}^h are homogeneous coordinates of P_{im}. If $f_u \neq 0$ and $f_v \neq 0$, the pixel (u, v) defines the relations $x_c/z_c, y_c/z_c$ (5). However, we need to know the value $z_c(u, v)$ ("depth

map") to recover original 3D coordinates x_x, y_c, and z_c from the pixel coordinates (u, v). Additional information about the scene is needed to build the depth map $z_c(u, v)$ for some range of pixels (u, v). Perspective projection preserves straight lines.

A wide-angle lens, used by most street public cameras, introduces perspective distortion described by the pinhole camera model and substantial radial distortion. Usually, the radial distortion is easily detectable, especially on "straight" lines near the edges of the image. An extended model was used to take it into account. A popular quadratic model of radial distortion ([6] (pp. 63–66), [7] (pp. 189–193), [5]) changes the pinhole camera model (6) as follows:

$$r^2 = (x_c/z_c)^2 + (y_c/z_c)^2, \tag{7}$$

$$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4) \begin{bmatrix} x_c/z_c \\ y_c/z_c \end{bmatrix}, \tag{8}$$

$$P^h_{im} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} \hat{x} \\ \hat{y} \\ 1 \end{bmatrix} = \begin{bmatrix} f_u \hat{x} + c_u \\ f_v \hat{y} + c_v \\ 1 \end{bmatrix}, \tag{9}$$

where k_1 and k_2 are radial distortion coefficients. In addition to the quadratic, more complex polynomials and models of other types are used ([5], [6] (pp. 63–66, 691–692), [7] (pp. 189–193)). Let the coefficients f_u, f_v, c_u, c_v, k_1, and k_2 be selected so that the formation of images from the camera is sufficiently well described by (7)–(9). To construct the artificial image on which the radial distortion k_1 and k_2 is eliminated, we must for each pair of relations $(x_c/z_c, y_c/z_c)$ obtain new positions of pixels $(\hat{u}, \hat{v})$ according to the pinhole camera model:

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \begin{bmatrix} f_u x_c/z_c + c_u \\ f_v y_c/z_c + c_v \end{bmatrix}. \tag{10}$$

To do this, we need all values $(x_c/z_c, y_c/z_c)$ (coordinates in F_c), which form the original image. To obtain the relations from an image, you have the image pixel coordinates (u, v) and equations (follows from (7)–(9))

$$Q(u, v) = \frac{f_v(u - c_u)}{f_u(v - c_v)}, \; D(u, v) = Q^2 + 1, \tag{11}$$

$$x_c/z_c = Q(u, v) y_c/z_c, \tag{12}$$

$$k_2 D(u, v)^2 (y_c/z_c)^5 + k_1 D(u, v)(y_c/z_c)^3 + y_c/z_c + \frac{c_v - v}{f_v} = 0, \tag{13}$$

if $v \neq c_v$. Similar equations are valid for $v = c_v$ and $u \neq c_u$. Equation (13) has five roots (complex, generally speaking), so we need an additional condition to select one root. We can select the real root nearest to $(v - c_v)/f_v$. The artificial image will not be a rectangle, but we can select a rectangle part of it (see figures of Example 2). Note that the mapping $(u, v) \to (\hat{u}, \hat{v})$ is determined by the camera parameters f_u, f_v, c_u, c_v, k_1, and k_2. It does not change from frame to frame and can be computed once and used before the parameters are changed.

3. Public Camera Parameters Estimation

Model parameters (3), (7)–(9) define the transformation of the 3D point, visible by camera, to pixel coordinates in the image. These parameters are: rotation matrix of the camera orientation R (2) and (3); camera position coordinates (point O) in F_{enu} (3); intrinsic camera parameters f_u, f_v, c_u, and c_v, (1) and (4); and radial distortion coefficients k_1 and k_2, (8) and (9).

The camera calibration process estimates the parameters using a set of 3D point coordinates $\{P^i_{enu}\}$ and a set of the point image coordinates $\{P^i_{im}\}$. The large set $\{P^i_{enu}\}$ is called a point cloud. There are software libraries that work with point clouds. Long range

3D lidars or geodesic instruments measure 3D coordinates values for a set $\{P^i_{enu}\}$. Another camera depth map can help obtain a set $\{P^i_{enu}\}$. Stereo/multi-view cameras can build depth maps, but it is hard to obtain high accuracy for long distances. When the tools listed above are unavailable, it is possible to obtain global coordinates of points in the camera field of view by GNSS sensors or from online maps. The latter variants are easier to access but less accurate. There are many tools [8] to translate global coordinates to an ENU.

Camera calibration procedure is well studied ([7] (pp. 178–194), [9] (pp. 22–28), [10], [6] (pp. 685–692)). It looks for the parameter values that minimize the difference between pixels $\{P^i_{im}\}$ and pixels generated by the model (3), (7)–(9) from $\{P^i_{enu}\}$. Computer vision software libraries include calibration functions for a suitable set of images from a camera [5]. The OpenCV function `calibrateCamera` [5,11] needs a set of points and pixels of a special pattern (e.g., "chessboard") rotated before the camera. The function returns an estimation of intrinsic camera parameters and camera placements concerning the pattern positions. These relative camera placements are usually useless after outdoor camera installation. If the camera uses a zoom lens and the zoom changes on the street, the camera focal length changes too, and new calibration is needed. If we use a public camera installed on a wall or tower, it is hard to collect appropriate pattern images from the camera to apply a function such as `calibrateCamera`. Camera operators usually do not publish camera parameters, but this information is indispensable for many computer vision algorithms. We need a calibration procedure applicable to the available online data.

Site-Specific Calibration

If the unified calibration procedure data are unavailable, we can estimate camera position and orientation parameters (R,O) separately from others. Resolution $N \times M$ of images/video from a camera is available with the images/video. As noted earlier, usually

$$c_u = N/2, c_v = M/2. \tag{14}$$

Many photo cameras (including phone cameras) add the EXIF metadata to the image file, which often contain focal length f. The image sensor model description can contain a pixel width w and height h, so (4) gives f_u and f_v values. The distortion coefficients are known for high-quality photo lenses. Moreover, the raw image processing programs can remove the lens radial distortion. If lucky, we can obtain intrinsic camera parameters and radial distortion coefficients and apply a pose computation algorithm (PnP, [12,13]) to sets $\{P_{enu}\}$, $\{P_{im}\}$ to obtain R,O estimates. When EXIF metadata from the camera are of no help (this is typical for most public cameras), small sets $\{P_{enu}\}$ and $\{P_{im}\}$ and Equation (5) can help to obtain f_u and f_v estimations. Suppose we know the installation site of the camera (usually, the place is visible from the camera's field of view). In that case, we can estimate GNSS/ENU coordinates of the place (point O coordinates) by the method listed earlier. The camera orientation (matrix R) can be detected with the point G coordinates (Figure 1) and the estimation of the camera horizon tilt. Horizontal or vertical 3D lines (which can be artificial) in the camera's field of view can help evaluate the tilt.

4. Formulation of the Problem and Its Solution Algorithm

Designers and users of transport models are interested in the flow density (number of vehicles per unit of lane length or direction at a time); speed of vehicles (on a lane or direction); and intensity of the flow (number of vehicles crossing the cross section of a lane or direction). We capture some areas (ROI) of the frames to determine these values from a fixed camera. The camera generates a series of ROI images, usually at a fixed interval, such as 25 frames/second. The algorithms (object detection or instance segmentation or contours detection and object tracking, see [14–17]) find a set of contours describing the trajectory of each vehicle crossing the ROI. The contour description consists of the coordinates of the vertices in F_{im}. We count the number of contours per meter in the ROI of each frame to estimate flow density. We choose the vertex (e.g., "bottom left") of the contour in the trajectory and count the meters that the vertex has passed in the trajectory to

estimate the car speed. Both cases require estimation in meters of distance given in pixels, so we need to convert lengths in the image (in F_{im}) to lengths in space (in F_{enu} or F_c). In some examples, distances in pixels are related to metric distances almost linearly (where radial and perspective distortions are negligible). We will consider the public cameras that produce images with significant radial and perspective distortions in the ROI (more common case).

Problem 1.

1. *Let Q be the area that is a plane section of the road surface in space (the road plane can be inclined);*
2. *Φ is the camera frame of $N \times M$ resolution containing the image of Q, denoting the image by Q_{im};*
3. *The camera forms pixel coordinates of the image Φ according to the model (3), (7)–(9) with unknown parameters f_u, f_v, k_1, k_2, R,*

$$Q_{im} = \Lambda(Q_{enu});$$

4. *Φ contains the image of at least one segment, the prototype of which in space is a segment of a straight line with a known slope (e.g., vertical);*
5. *The image is centered relative to the optical axis of the video camera, that is, $c_u = N/2$ and $c_v = M/2$;*
6. *The F_{enu} coordinates of the points O (camera position) and G (the source of the optical center C) are known;*
7. *The F_{im} coordinates of one or more pixels of $\{P^u_{im} \neq C\}$ located on the line $v = c_v$ and the F_{enu} coordinates of their sources $\{P^u_{enu}\}$ are known;*
8. *The F_{im} coordinates of one or more pixels of $\{P^v_{im} \neq C\}$ located on the line $u = c_u$ and the F_{enu} coordinates of their sources $\{P^v_{enu}\}$ are known;*
9. *The F_{enu} coordinates of three or more points $\{P^Q_{enu}\} \in Q$ are known; at least three of them must be non-collinear;*
10. *The F_{im} coordinates of one or more groups of three pixels $\{(P^a_{im}, P^b_{im}, P^c_{im})\}$ are known, and in the group, the sources of the pixels are collinear in space.*

Find the parameters of the camera f_u, f_v, R, k_1, and k_2 and construct the mapping

$$Q_{enu} = \Lambda^{-1}(Q_{im}).$$

Online maps allow remote estimation of global coordinates of points and horizontal distances. Many such maps do not show the altitude, and most do not show the height of buildings, bridges, or vegetation. Online photographs, street view images, and horizontal distances can help estimate such objects' heights. Camera locations are often visible in street photos. This variant of measurements suggests that the coordinate estimates may contain a significant error. The errors result in some algorithms (e.g., PnPransac) being able to generate a response with an unacceptable error.

To find $\{P^u_{im}\}$,$\{P^u_{enu}\}$,$\{P^v_{im}\}$, and $\{P^v_{enu}\}$ coordinates, the points must be visible both in the image and on the online map.

4.1. Solution Algorithm

We want to eliminate the radial distortion of area Q_{im} to go to the pinhole camera model. From (10)–(13), it follows that for this you need values f_u and f_v.

4.1.1. Obtain the Intrinsic Parameters (Matrix A)

Note that from $v^u = c_v$ and (7)–(9) it follows $yc/zc = 0$ for the point $P^u_{im} = (u^u, v^u)$ (because $1 + k_1 r^2 + k_2 r^4 = 0$ leads to $u^u = c_u$ and $P^u_{im} = C$). So, the point P^u_{im} stays on the central horizontal line of the image for any values k_1 and k_2. By analogy, for $P^v_{im} = (u^v, v^v)$ take place $x_c/z_c = 0$, and the point stays on the central vertical line for any values k_1 and k_2.

Evaluate the angles between the optical axis OG and vectors OP^u_{enu} and OP^v_{enu} (Figure 2, Figure 1):

$$\begin{array}{l} \theta_x = \arccos\left(OG \cdot OP^u_{enu}/(|OG||OP^u_{enu}|)\right) \\ \theta_y = \arccos\left(OG \cdot OP^v_{enu}/(|OG||OP^v_{enu}|)\right) \end{array}. \quad (15)$$

It follows from (5) that if the effect of radial distortion on the values (u^u, v^u) and (u^v, v^v) can be ignored (camera radial distortion is moderate, and the points are not too far from C), then

$$\begin{array}{l} f_u = (u^u - c_u)/\tan\theta_x \\ f_v = (v^v - c_v)/\tan\theta_y \end{array}. \quad (16)$$

Equation (16) gives the initial approximation of coefficients f_u and f_v and an evaluation of matrix A.

4.1.2. Obtain the Radial Distortion Compensation Map

Let $\hat{\Phi}$ be the image obtained from Φ by the transformation (10) and solution of Equations (11)–(13). Denote Θ the mapping of Φ to $\hat{\Phi}$:

$$\hat{\Phi} = (\hat{u}, \hat{v}) = \Theta(u, v; k_1, k_2, f_u, f_v, c_u, c_v) = (f_u x_c/z_c + c_u, f_v y_c/z_c + c_v). \quad (17)$$

We create the image $\hat{\Phi}$ according to the pinhole camera model. The model transforms a straight line in space into a straight line in the image.

Let $P^a_{im} = (u^a, v^a)$, $P^b_{im} = (u^b, v^b)$, and $P^c_{im} = (u^c, v^c)$ in F_{im} and

$$\hat{P}^a_{im} = (\hat{u}^a, \hat{v}^a) = \Theta(u^a, v^a), \hat{P}^b_{im} = (\hat{u}^b, \hat{v}^b) = \Theta(u^b, v^b), \hat{P}^c_{im} = (\hat{u}^c, \hat{v}^c) = \Theta(u^c, v^c). \quad (18)$$

We can find k_1 and k_2 values that minimize the sum of distances from pixels $\hat{P}^b_{im}$ to the lines passing through $\hat{P}^a_{im}$ and $\hat{P}^c_{im}$. We calculate the distance as:

$$\frac{|(\hat{u}^c - \hat{u}^a)(\hat{v}^a - \hat{v}^b) - (\hat{u}^a - \hat{u}^b)(\hat{v}^c - \hat{v}^a)|}{\sqrt{(\hat{u}^a - \hat{u}^c)^2 + (\hat{v}^a - \hat{v}^c)^2}} \quad (19)$$

for each triplet $(P^a_{im}, P^b_{im}, P^c_{im})$. This approach is a variant of the one described, for example, in [7] (pp. 189–194). OpenCV offers the realization of Θ^{-1} (`initUndistortRectifyMap`). It is fast, but we need to invert it for our case. We solve Equations (11)–(13) (including version for $v = c_v, u \neq c_u$) to obtain Θ. The Θ^{-1} is polinomial from $\hat{u}, \hat{v}$ (see (7)–(10)).

4.1.3. Obtain the Camera Orientation (Matrix R)

To determine the camera orientation, we use the point O and the point G given in the coordinates F_{enu} (Figure 1). Unit vector

$$\vec{\mathbf{e}}_{zc} = OG/|OG| = (e^{zc}_x, e^{zc}_y, e^{zc}_z)_{enu} \quad (20)$$

gives direction to axis Z_c of frame F_c. $\vec{\mathbf{e}}_{zc}$, and O determines the plane of points x, for which the vector $\vec{Ox}$ is perpendicular to $\vec{\mathbf{e}}_{zc}$. In this plane, lie the axes X_c and Y_c of coordinate frame F_c. Unit vector $\vec{\mathbf{e}}_d = (0, 0, -1)_{enu}$ is downward. Let F_v be camera coordinates with the same optical axis Z_c as F_c, but the F_v has zero horizon tilt. Axis X_v of the frame is perpendicular to $\vec{\mathbf{e}}_d$ as far as X_v is parallel to the horizontal plane. We can find axes directions Y_v and X_v of the frame F_v:

$$\begin{array}{l} \vec{\mathbf{n}}_{yv} = \vec{\mathbf{e}}_d - (\vec{\mathbf{e}}_d \cdot \vec{\mathbf{e}}_{zc})\vec{\mathbf{e}}_{zc} = \vec{\mathbf{e}}_d + e^{zc}_z \vec{\mathbf{e}}_{zc} \\ \vec{\mathbf{e}}_{yv} = \vec{\mathbf{n}}_{yv}/|\vec{\mathbf{n}}_{yv}| \\ \vec{\mathbf{e}}_{xv} = \vec{\mathbf{e}}_{yv} \times \vec{\mathbf{e}}_{zc} \end{array}. \quad (21)$$

Vectors (21) form the orthonormal basis, which allows us to construct a rotation matrix R_{v2e} for transition from F_v to F_{enu}:

$$R_{v2e} = \begin{bmatrix} e_x^{xv} & e_x^{yv} & e_x^{zc} \\ e_y^{xv} & e_y^{yv} & e_y^{zc} \\ e_z^{xv} & e_z^{yv} & e_z^{zc} \end{bmatrix} = \begin{bmatrix} \vec{\mathbf{e}}_{xv} & \vec{\mathbf{e}}_{yv} & \vec{\mathbf{e}}_{zc} \end{bmatrix}. \tag{22}$$

If R_{v2e} is a rotation matrix, the following equations hold:

$$R_{e2v} = R_{v2e}^{-1} = R_{v2e}^{T}. \tag{23}$$

From Clause 4 of the problem statement, there is a line segment in the artificial image $\hat{\Phi}$ with a known slope in space. It is possible to compare the segment slope in the image $\hat{\Phi}$ and the slope of its source in space. So, we can estimate the camera horizon tilt angle (denote it as γ) and rotate the plane with the axes X_v and Y_v around the optical axis Z_c at this angle. The resulting system of coordinates F_c corresponds to the actual camera orientation. To pass from the camera coordinates F_c to F_{enu}, we can first go from F_c to F_v by rotation with the angle γ around the optical axis Z_c using the rotation matrix

$$\Gamma = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{24}$$

We can describe the transition from F_c to F_{enu} (without origin displacement) as a combination of rotations by the matrix

$$R_{c2e} = R_{v2e}\Gamma, \tag{25}$$

which is also a rotation matrix. The matrix that we have already designated R (2) gives the inverse transition from F_{enu} to the coordinates of the camera frame F_c :

$$R = R_{c2e}^{T} \tag{26}$$

and the shift t, which in the coordinates F_c characterizes the transition from the beginning of the coordinates of F_{enu} to the point of installation of the camera O. The t is often more easily expressed through the coordinates O_{enu} given in F_{enu}, see (3). To convert the coordinates of a P_{enu} from F_{enu} to P_c coordinates for the F_c camera frame, use the following expressions:

$$P_c = R(P_{enu} - O) = RP_{enu} + t. \tag{27}$$

Typically, an operator aims to set a camera with zero horizon tilt.

4.1.4. Obtain the Mapping Λ^{-1} for Q_{im}

Let $\hat{Q}_{im} = \Theta(Q_{im})$ be the area in $\hat{\Phi}$ corresponding to Q_{im} in the image Φ. Q_{enu} and Q_c are the domain Q on the road plane in coordinates F_{enu}, and F_c, respectively. From (27), we obtain

$$Q_{enu} = R^T Q_c + O,\ Q_c = R(Q_{enu} - O). \tag{28}$$

From Clause 2 of the problem statement, it follows that an image of Q_c is visible in Φ (and in $\hat{\Phi}$, so $\hat{Q}_{im} \subset \hat{\Phi}$).

We can convert the ENU coordinates of points $\{P_{enu}^{Q}\}$ to F_c by following (27). We denote the result as $\{P_c^{Q_i}\}_{i=1}^{L}$. Let $P_c^{Q_i} = (x_c^i, y_c^i, z_c^i)$.

We approximate its plane with the least squares method. The plane is defined in F_c by the equation

$$p_x x_c + p_y y_c + z_c = p_z,\ p = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix}. \tag{29}$$

The matrix D and vector E represent the points on the road:

$$D = \begin{bmatrix} x_c^1 & y_c^1 & -1 \\ x_c^2 & y_c^2 & -1 \\ \vdots & \vdots & \vdots \\ x_c^L & y_c^L & -1 \end{bmatrix}, E = \begin{bmatrix} -z_c^1 \\ -z_c^2 \\ \vdots \\ -z_c^L \end{bmatrix}. \tag{30}$$

The plane parameters p can be found by solving the least squares problem

$$\min_q \|Dq - E\|^2 = \|Dp - E\|^2, \tag{31}$$

the exact solution to the least squares task is:

$$p = (D^T D)^{-1} D^T E. \tag{32}$$

We denote by Π the plane defined by (29) and, (32). Note that $Q_c \subset \Pi$.

For a point $(x_c, y_c, z_c) \in \Pi$ represented by the pixel coordinates $(\hat{u}, \hat{v})$ in $\hat{\Phi}$ we obtain, taking into account (6)

$$\begin{array}{rcl} z_c & = & p_z - p_x x_c - p_y y_c \\ x_c/z_c & = & (\hat{u} - c_u)/f_u \\ y_c/z_c & = & (\hat{v} - c_v)/f_v \end{array} \tag{33}$$

.

If $\hat{u} \neq c_u$ and $\hat{v} \neq c_v$ (it means that $x_c \neq 0$ and $y_c \neq 0$) then

$$\begin{array}{rcl} p_z - p_x x_c - p_y y_c & = & x_c f_u/(\hat{u} - c_u) \\ p_z - p_x x_c - p_y y_c & = & y_c f_v/(\hat{v} - c_v), \end{array} \tag{34}$$

so there are linear equations that allow us to express x_c and y_c through $\hat{u}$ and $\hat{v}$. Let

$$\begin{array}{l} a(\hat{u}) = f_u/(\hat{u} - c_u) + p_x \\ b(\hat{v}) = f_v/(\hat{v} - c_v) + p_y \end{array}. \tag{35}$$

From (34), we obtain solution:

$$\begin{bmatrix} x_c(\hat{u}, \hat{v}) \\ y_c(\hat{u}, \hat{v}) \end{bmatrix} = \begin{cases} \begin{bmatrix} p_z(p_y - b(\hat{v}))/(p_x p_y - a(\hat{u})b(\hat{v})) \\ p_z(p_x - a(\hat{u}))/(p_x p_y - a(\hat{u})b(\hat{v})) \end{bmatrix}, & \text{if } \hat{u} \neq c_u, \hat{v} \neq c_v \\ \begin{bmatrix} 0 \\ p_z/b(\hat{v}) \end{bmatrix}, & \text{if } \hat{u} = c_u, \hat{v} \neq c_v \\ \begin{bmatrix} p_z/a(\hat{u}) \\ 0 \end{bmatrix}, & \text{if } \hat{u} \neq c_u, \hat{v} = c_v \\ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, & \text{if } \hat{u} = c_u, \hat{v} = c_v \end{cases}, \tag{36}$$

where $a(\hat{u})$ and $b(\hat{v})$ defined in (35) and

$$z_c(\hat{u}, \hat{v}) = p_z - p_x x_c(\hat{u}, \hat{v}) - p_y y_c(\hat{u}, \hat{v}). \tag{37}$$

We have constructed a function that assigns to each pixel coordinates $g = (\hat{u}, \hat{v})$ (in image $\hat{\Phi}$) a spatial point in the coordinate system F_c, according to (36) and (37):

$$\xi(g) = \begin{bmatrix} x_c(\hat{u}, \hat{v}) \\ y_c(\hat{u}, \hat{v}) \\ z_c(\hat{u}, \hat{v}) \end{bmatrix}. \tag{38}$$

If $g \in \hat{Q}_{im}$ $\xi(g) \in Q_c$, so

$$Q_c = \xi(\hat{Q}_{im}), \tag{39}$$

$$Q_{enu} = R^T Q_c + O = R^T \xi(\hat{Q}_{im}) + O = R^T \xi(\Theta(Q_{im})) + O = \Lambda^{-1}(Q_{im}). \tag{40}$$

Next, we will refer more to (39). F_{enu} helps obtain measurement results, but F_c is sufficient for calculating metric lengths. There are other ways to map pixels to meters (see, e.g., [7] (pp. 47–55)), but their applicability depends on the data available.

4.2. Auxiliary Steps

4.2.1. Describing of the Area Q

Different Q shapes help for varying tasks. We fix $\hat{Q}_{im}$ as a quadrilateral on the image $\hat{\Phi}$. The Q_c (source of the quadrilateral in space) in the plane Π may be a rectangle or a tetragon. We choose the four corners of g_1, g_2, g_3, and g_4 of the domain $\hat{Q}_{im}$ as pixels in the $\hat{\Phi}$. The "tetragon" $Q_{im} = \Theta^{-1}(\hat{Q}_{im})$ is the ROI in the original image Φ, in which we use object detection to estimate transport flows statistics. The lines bound $\hat{Q}_{im}$ are ([7] (p. 28)):

$$l^h_{bl} = g^h_1 \times g^h_2,\ l^h_{tl} = g^h_1 \times g^h_3,\ l^h_{tr} = g^h_3 \times g^h_4,\ l^h_{br} = g^h_2 \times g^h_4, \tag{41}$$

where $g^h_i = (\hat{u}^i, \hat{v}^i, 1)$ are the homogeneous coordinates of pixel $g_i = (\hat{u}^i, \hat{v}^i)$. Pixel $g^h \in \hat{Q}_{im}$ must be the solution to a system of four inequalities:

$$\begin{cases} l^h_{tl} \cdot g^h \geq 0 \\ l^h_{tr} \cdot g^h \geq 0 \\ l^h_{bl} \cdot g^h \leq 0 \\ l^h_{br} \cdot g^h \leq 0 \end{cases}. \tag{42}$$

The Q_{im} is a continuous domain in $\mathbb{R}^2$, but the whole image Φ contains only a fixed number of actual pixels ($N \times M$). The same is true for $\hat{Q}_{im} = \Theta(Q_{im})$. We can obtain real pixels that fall in Q_{im} from $\hat{Q}_{im}$ with the per-pixel correspondence $Q_{im} = \Theta^{-1}(\hat{Q}_{im})$. We apply the mapping (39) and obtain the set of points in F_c which correspond to actual pixels in Q_{im}. F_c is a metric coordinate system, so the distance in F_c can be used to estimate meters/second or objects/meter. The same is true for any rotations or shifts of the frame F_c.

4.2.2. Coordinate Frame Associated with the Plane Π

We can go from F_c to a coordinate system associated with the plane Π. We apply it for illustrations, but it can be helpful for other purposes. We can regard the traffic in Q_{im} as anything that rises above the plane Π . We denote by F_π the coordinate system connected to Π. and use the normal to the plane Π at the coordinates F_c from (32) as the axis Z_π.

$$\vec{\mathbf{n}}_\pi = \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix},\ \vec{\mathbf{e}}_{z\pi} = \vec{\mathbf{n}}_\pi / |\vec{\mathbf{n}}_\pi|. \tag{43}$$

We choose in the Π plane the direction of another axis (e.g., Y_π), and the third axis is determined automatically. Let the Y axis be pointed by the $g_4 - g_2$ vector (that is, along the direction of traffic movement in the Q area), then

$$\begin{matrix} \vec{\mathbf{e}}_{yp} = & (\xi(g_4) - \xi(g_2))/|\xi(g_4) - \xi(g_2)| \\ \vec{\mathbf{e}}_{xp} = & \vec{\mathbf{e}}_{yp} \times \vec{\mathbf{e}}_{zp} \end{matrix}. \tag{44}$$

Select the origin of the coordinate system F_π as follows

$$o = \xi(g_2). \tag{45}$$

The rotation matrices from F_p to F_c and their inverses look like the following:

$$R_{\pi 2c} = \left[\vec{\mathbf{e}}_{x\pi}, \vec{\mathbf{e}}_{y\pi}, \vec{\mathbf{e}}_{z\pi}\right] = \begin{bmatrix} e_x^{x\pi} & e_x^{y\pi} & e_x^{z\pi} \\ e_y^{x\pi} & e_y^{y\pi} & e_y^{z\pi} \\ e_z^{x\pi} & e_z^{y\pi} & e_z^{z\pi} \end{bmatrix}, \; R_{c2\pi} = R_{\pi 2c}^T. \tag{46}$$

Use the following expression for the translation of coordinates of P_c given in frame F_c to P_π given in frame F_π:

$$P_\pi = R_{c2\pi}(P_c - o). \tag{47}$$

5. Examples

Consider a couple of public camera parameters evaluations, where the images and small sets of coordinates $\{P_{enu}\}$ and $\{P_{im}\}$ of limited accuracy are available.

5.1. Example 1: Inclined Driving Surface and a Camera with Tilted Horizon

Figure 3 shows a video frame from the public camera.

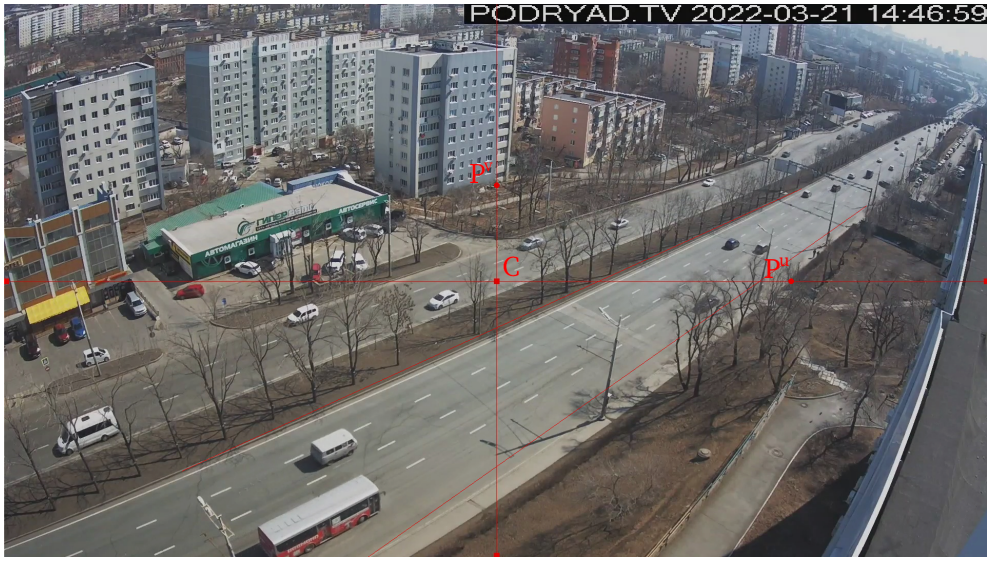

Figure 3. Field of view of the public street camera in Vladivostok. Example 1, image Φ.

The camera is a good example, as its video contains noticeable perspective and radial distortion. In the image, there is a line demonstrating the camera's slight horizon tilt (the wall at the base of which is P^v). The visible part of the road has a significant inclination. This is a FullHD camera ($N = 1920, M = 1080$). Select a point P^0 as the origin of F_{enu}. Assess ENU coordinates of the point O (the camera position), point G (the source of the principal point C, Figures 1 and 3), and points P^u and P^v on the lines $v = c_v$ and $u = c_u$, respectively. Using maps and online photos, we obtained the values listed in Table 1. Convert global coordinates to the F_{enu} coordinates (see [8]) and add it to the table (in meters).

Table 1. The points used for the calibration. Latitude and longitude in degrees, altitudes and ENU coordinates in meters, coordinates of pixels in F_{im} units (u is the column index, v is the row).

Name	Lat	Long	Height	u	v	x_{enu}	y_{enu}	z_{enu}
P^0	43.175553°	131.917725°	56			0	0	0
G	43.176295°	131.918380°	57	960	540	53.3	82.4	1
O	43.176934°	131.917912°	98			15.2	153.4	42
P^u	43.176033°	131.917937°	57	1534	540	17.2	53.3	1
P^v	43.175828°	131.918728°	52	960	353	81.6	30.6	−4

The points from Table 1 can be found on the satellite layer of [18] using the latitude and longitude of the query.

We obtain $f_u = 1166.2$ and $f_v = 1241.55$ by using (15), (16) and Table 1 data. Street camera image sensors usually have square pixels, so $w = h$ and $f_u = f_v$ (4). If the difference f_u and f_v is small, let f_u, f_v be equal $f_u = f_v = 1203.89$ (we use mean value). So we have an approximation of matrix A.

Now, we can estimate radial distortion coefficients k_1 and k_2 by minimizing distances (19) or in another way. Put $k_1 = -0.24$ and $k_2 = 0$ to compute the mapping Θ (17) and apply it to eliminate radial distortion k_1 and k_2 from the original image (Φ). The mapping does not change for different frames from the camera video. We obtained the undistorted version of image Φ, which we have identified $\hat{\Phi}$ (Figure 4).

Figure 4. The image after compensation of radial distortion. Example 1, image $\hat{\Phi} = \Theta(\Phi)$.

The radial distortion of the straight lines in the vicinity of the road has almost disappeared in $\hat{\Phi}$. The camera's field of view decreased, the C point remained in place, and the p_1 and p_2 points moved further along the lines $v = c_v$ and $u = c_u$. The values of f_u and f_v can be recalculated, but radial distortion is not the only cause of errors. Therefore, we will perform additional cross-validation and compensate the values of f_u and f_v if required.

We can estimate the horizon tilt angle from the image (Figure 4) and rotate the plane with the axes X_v and Y_v of frame F_v around the optical axis Z_c at this angle. As a result of rotating the image around the optical axis of the camera on 4.1°, the verticality of the required line was achieved (Figure 5), so let $\gamma = 4.1°$ in (24).

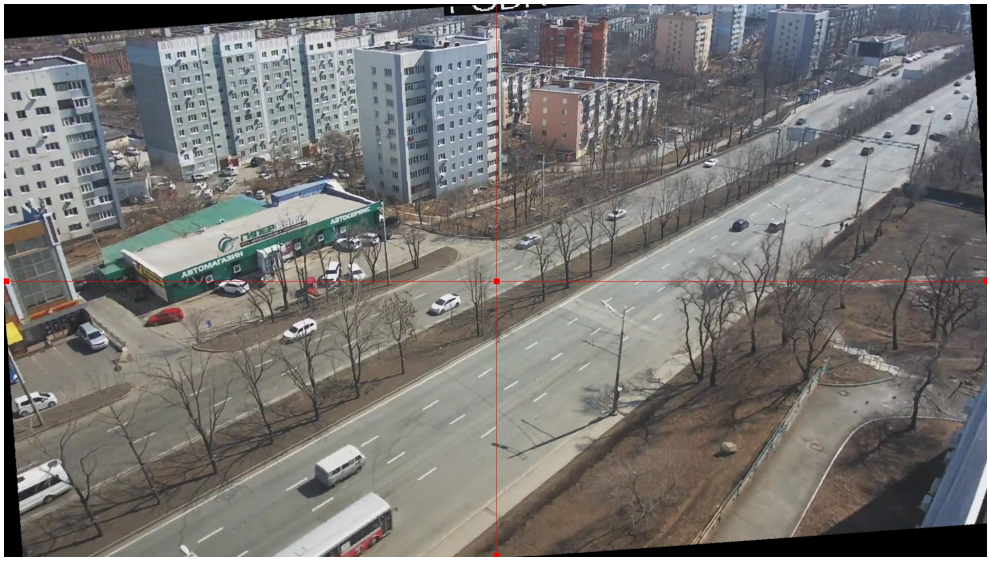

Figure 5. The image $\hat{\Phi}$ rotated by 4.1° around the optical axis of the camera.

We compute the camera orientation matrix R with (20)–(26). Now, we can convert F_{enu} coordinates to F_c with (27).

Let us use the area nearest to the camera carriageway region as the ROI (area Q). We select several points in Q, estimate their global coordinates, and convert them to F_{enu} [8]. Next, we convert F_{enu} coordinates to F_c with (27). The results are in Table 2.

Table 2. The spatial points used for the road plane approximation (global and F_c coordinates); F_c coordinates are in meters.

Num	Lat	Long	Height	x_c	y_c	z_c
1	43.176500°	131.918103°	59.36	8.22	12.45	61.94
2	43.176442°	131.918310°	59	−3.12	5.76	74.25
3	43.176532°	131.918362°	59.3	−11.75	7.97	68.04
4	43.176329°	131.918286°	58.6	4.82	2.07	83.47
5	43.176139°	131.917970°	57	37.46	2.9	89.96
6	43.176093°	131.918187°	56.9	24.76	−3.76	101.44
7	43.175553°	131.917725°	56	87.14	−14.46	133.16
8	43.175495°	131.917991°	55.9	71.67	−22.71	147.37
9	43.175794°	131.917845°	56	65.33	−7.39	116.24

We obtain et the road plane Π approximation with the (29) and (32):

$$p = (-0.20316, 2.04433, 86.99813). \tag{48}$$

We choose four corners of g_1, g_2, g_3 and g_4 of the domain $\hat{Q}_{im}$ in the $\hat{\Phi}$ (see Table 3 and Figure 6).

Table 3. Domain $\hat{Q}_{im}$ corners coordinates in $\hat{\Phi}$.

Corner	Name	u	v
top left	g_1	504	849
bottom left	g_2	739	1079
top right	g_3	1468	410
bottom right	g_4	1642	462

We calculate the lines that bound domain $\hat{Q}_{im}$ by (41) and detect the set of pixel coordinates that belong to $\hat{Q}_{im}$. Note that this set does not change for different frames from the camera video. We can save it for later usage with the camera and the Q.

We compute the Q_c as $\xi(\hat{Q}_{im})$ by (48) and (35)–(39). We convert the pixel coordinates from Q_c to F_π by (43)–(47):

$$Q_\pi = R_{c2\pi}(Q_c - o) \tag{49}$$

and save the result. The coordinates set Q_π does not change for the (fixed) camera and the Q. The obtained discrete sets Q_c and Q_π are sets of metric coordinates. We can use Q_π for measurements on the plane Π as is or apply an interpolation.

Since $Q_\pi \subset \Pi$, the point set is suitable to output in a plane picture. If for each point, we use $\xi(g) \in Q_\pi$ and the color of the pixel g for all $g \in \hat{Q}_{im} \subset \hat{\Phi}$, and output the plane as the scatter map, we obtain the bottom image from Figure 6. We chose a multi-car scene in the center of Q for the demonstration. We obtain the accurate "view from above" for the points that initially lie on the road's surface. The positions of the pixels representing objects that tower above plane Π have shifted. There are options for estimating and accounting for the height of cars and other objects so that their images look realistic in the bottom image. However, this is the subject of a separate article.

It is worth paying attention to the axes of the bottom figure (they show meters). In comparison with the original (top image of Figure 6), there are noticeable perspective distortion changes in the perception of distance. The width and length of the area are in good agreement with the measurements made by the rangefinder and estimates from online maps. The horizontal lines in the image are almost parallel which indicates the good quality of f_u and f_v. In this way, we cross-checked the camera parameters obtained earlier. Obviously, all vehicles are in contact with the road surface.

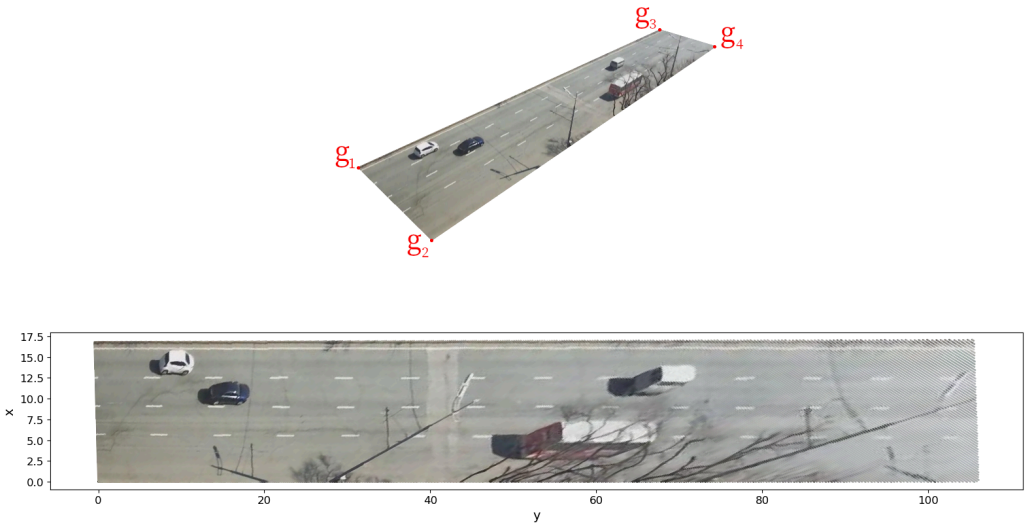

Figure 6. The Q_π set visualization in plane Π, by the colors of the pixels from $\hat{Q}_{im}$. Given the geometry of the sample, y_π is the horizontal axis, x_π the vertical.

We do not need to remove distortions (radial and perspective) from all video frames to estimate traffic statistics. The object detection or instance segmentation works with original video frames in Q_{im}. We use the distortions compensation for traffic measurements to obtain distances in meters for selected pixels. We calculate the needed maps once for the camera parameters and Q and use them as needed for measurements. Let us demonstrate this on a specific trajectory.

The detected trajectory of the vehicle consists of 252 contours, 180 of which are in Q_{im}. A total of 200 contours in the image look messy, so we draw every 20th (Figure 7). We deliberately did not choose rectangles to demonstrate a more general case. Vertex coordinates describe the contour. It is enough to select one point on or inside each contour to evaluate the speed or acceleration of an object. The point should not move around the object; we want the point source closer to the road's surface. The left bottom corner of a

contour is well suited for this camera. However, what does it mean for a polygon? For a contour $\{(u_i, v_i)\}_i$ we can build the pixel coordinates

$$(u_{min}, v_{max}) = (\min_i u_i, \max_i(v_i)).$$

Figure 7. Every twentieth contour of a vehicle's trajectory in the original image Φ.

We call the left bottom corner of a contour the vertex (u_j, v_j) for that

$$j = \arg\min_i |(u_i, v_i) - (u_{min}, v_{max})|,$$

considering the axes direction of the F_{im}. Another option is to search the point nearest to (u_{min}, v_{max}) on the edges of the contour with the help of (19).

We map all contour vertexes on the Q_π to obtain Figure 8. However, we need to map only these "corners" for measurements.

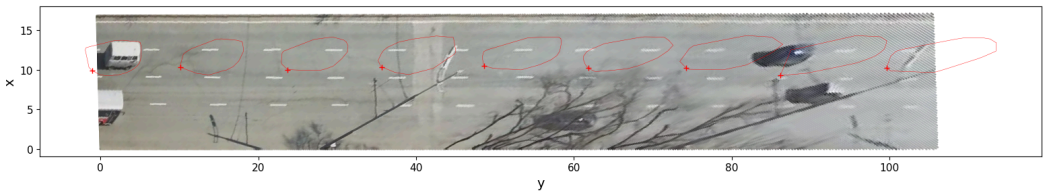

Figure 8. Every twentieth contour in the vehicle's trajectory mapped on the Q_π with the selected points.

The "corners" of the selected contours have the following y_π coordinates after the mapping:

$$\{-1.05, 10.13, 23.72, 35.66, 48.63, 61.80, 74.20, 86.15, 99.6\}.$$

We can use a variety of formulas to estimate speed, acceleration, and variation. The simplest estimation of the vehicle speed is (the camera frame rate is 25 frames/second, we use 1/20 of the frames):

$$= (25/20) * (99.6 + 1.05)/8 = 15.72656 \text{ m/s}$$

or 56.6 km/h.

5.2. Example 2. More Radial Distortion and Vegetation, More Calibration Points

Figure 9 illustrates a frame image from another public camera of the same operator. This camera has zero horizon tilt and more substantial radial distortion. We have calibrated the camera and calculated the maps in the season of rich vegetation. Tree foliage complicates the selection and estimation of coordinates of points. The two-way road is visible to the camera. This is a FullHD camera ($N = 1920, M = 1080, c_u = N/2, c_v = M/2$). The zero horizon tilt is visible on the line $u = c_u$ near P_5^v (see Table 4). We select the point P^0 as the origin of F_{enu} (see Table 4). We assess global coordinates of the point O (the camera position), point G (the source of the principal point C, Figure 1, Figure 9), and points $P_1^u, P_2^v, P_3^u, P_4^v$, and P_5^v on the lines $v = c_v$ (index u) or $u = c_u$ (index v). Next, we convert the global coordinates to F_{enu} and append them to Table 4.

We obtained estimations $f_u^1 = 1161.93, f_v^2 = 1382.45, f_u^3 = 1168.78, f_v^4 = 1354.82$, and $f_v^5 = 1349.67$ using Formulas (15) and (16) and Table 4 data. There are a few points, but one can assume a pattern $f_v > f_u$. Perhaps the camera's sensor pixel has a rectangular shape (see (4)). Consider the two hypotheses:

$$f_u = (f_u^1 + f_u^3)/2 = 1165.355, f_v = (f_v^2 + f_v^4 + f_v^5)/3 = 1362.313; \quad (50)$$

$$f_u = f_v = (f_u^1 + f_u^3 + f_v^2 + f_v^4 + f_v^5)/5 = 1283.53. \quad (51)$$

The second variant means the square pixel.

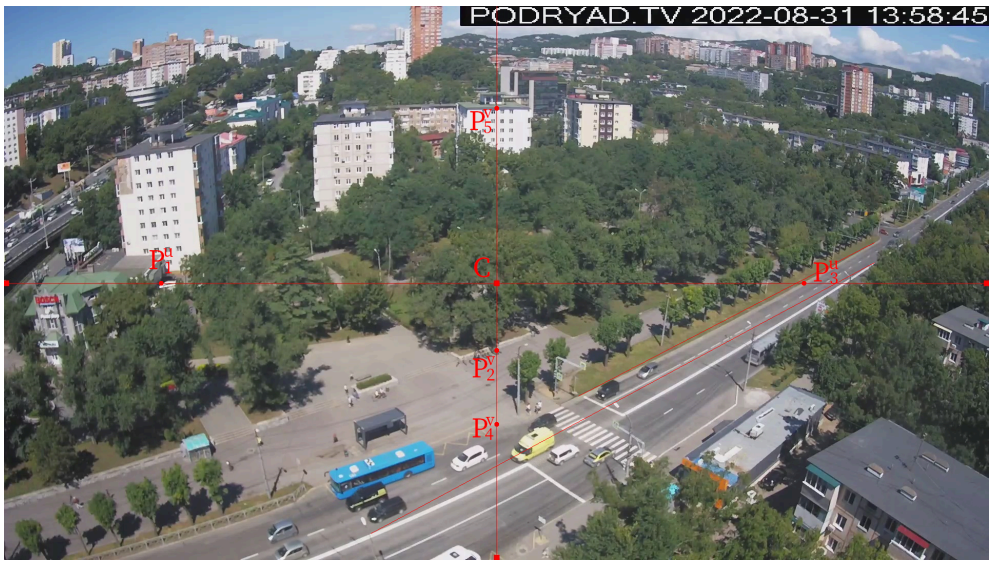

Figure 9. Field of view of the public street camera in Vladivostok. Example 2, image Φ.

Table 4. The points used for the calibration. Latitude and longitude are in degrees, altitudes and ENU coordinates are in meters, and coordinates of pixels are in F_{im} units (u is the column index, v is the row).

Name	Lat	Long	Height	u	v	x_{enu}	y_{enu}	z_{enu}
P^0	43.168143°	131.916257°	14			0	0	0
G	43.168356°	131.916733°	25	960	540	38.7	23.66	11
O	43.167616°	131.916245°	56			−0.97	−58.54	42
P_1^u	43.168775°	131.916107°	18	306	540	−12.2	70.2	4
P_2^v	43.168361°	131.916740°	15	960	671	39.27	24.22	0.7
P_3^u	43.168330°	131.917431°	14	1559	540	116.61	25.89	0
P_4^v	43.168200°	131.916631°	13.5	960	815	30.41	6.33	0
P_5^v	43.169134°	131.917249°	43	960	199	80.67	110.1	27

Let us try variant (50). With f_u and f_v we obtained the intrinsic parameters matrix A (1). We estimate radial distortion coefficients k_1 and k_2, put $k_1 = -0.17$ and $k_2 = 0.01$ to compute the mapping Θ (17), and apply it to eliminate radial distortion from the (Φ). We obtained the undistorted image $\hat{\Phi}$ (see Figure 10). We showed a cropped and interpolated version of Φ in Example 1 (Figure 4). Let us demonstrate the result of mapping Θ without postprocessing. The image's resolution $\hat{\Phi}$ is 2935×1651, but it contains only 1920×1080 pixels colored by the camera. So, the black grid is the unfilled pixels of the large rectangular.

Figure 10. The image Φ after compensation of radial distortion; Example 2, image $\hat{\Phi} = \Theta(\Phi)$ without postprocessing.

Let us return to the illustration that is more pleasing to the eye by cutting off part of $\hat{\Phi}$ and filling the void with interpolation (see Figure 11).

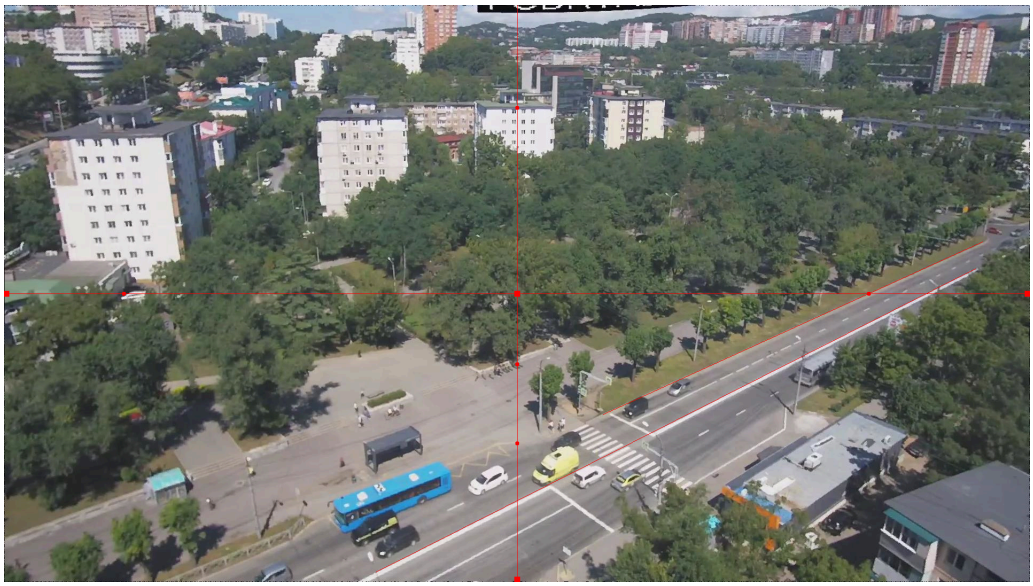

Figure 11. Cropped and interpolated part of image $\hat{\Phi}$.

We noted that in the field of view of the camera, there is a section of the wall of the building (near P_5^u), allowing you to put $\gamma = 0$ (see Figure 9). We are ready to compute the orientation matrix R (20)–(26), so we can convert F_{enu} coordinates to F_c (27). We select the points to approximate the road surface (see Table 5).

Table 5. The points used to approximate road plane Π (global and F_c coordinates, F_c and altitudes given in meters).

Num	Lat	Long	Height	x_c	y_c	z_c
1	43.168145°	131.916262°	13.5	0	0	−0.1
2	43.168192°	131.916673°	13.5	33.82	5.44	−0.1
3	43.168275°	131.917036°	14	63.34	14.67	−0.1
4	43.168367°	131.917515°	14.3	102.29	24.89	−0
6	43.168519°	131.918425°	15	176.29	41.77	0.1
7	43.167925°	131.916348°	14	7.4	−24.22	−0.1
8	43.168005°	131.916540°	14	23.01	−15.33	−0.1
9	43.168011°	131.916699°	14	35.94	−14.66	−0.1
10	43.168110°	131.917060°	14	65.3	−3.67	−0.1
11	43.168210°	131.917540°	14.5	104.33	7.44	−0.1

We calculate road plane Π with (29) and (32):

$$p = (0.00006008, 2.93113699, 130.65768737). \tag{52}$$

We choose four corners of g_1, g_2, g_3, and g_4 of $\hat{Q}_{im}$ in the image $\hat{\Phi}$ (see Table 6 and Figure 12). Let us try the nonrectangular area $\hat{Q}_{im}$.

Table 6. Domain $\hat{Q}_{im}$ corners coordinates in $\hat{\Phi}$, Example 2.

Corner	Name	u	v
top left	g_1	509	962
bottom left	g_2	993	1078
top right	g_3	1630	498
bottom right	g_4	1858	558

We calculate the lines that bound domain $\hat{Q}_{im}$ with (41) and detect the set of pixel coordinates that belongs to $\hat{Q}_{im}$. We compute the Q_c as $\xi(\hat{Q}_{im})$ with (48) and (35)–(39). Next, we convert the pixel coordinates from Q_c to F_π with (43)–(47) and (49) and save the result.

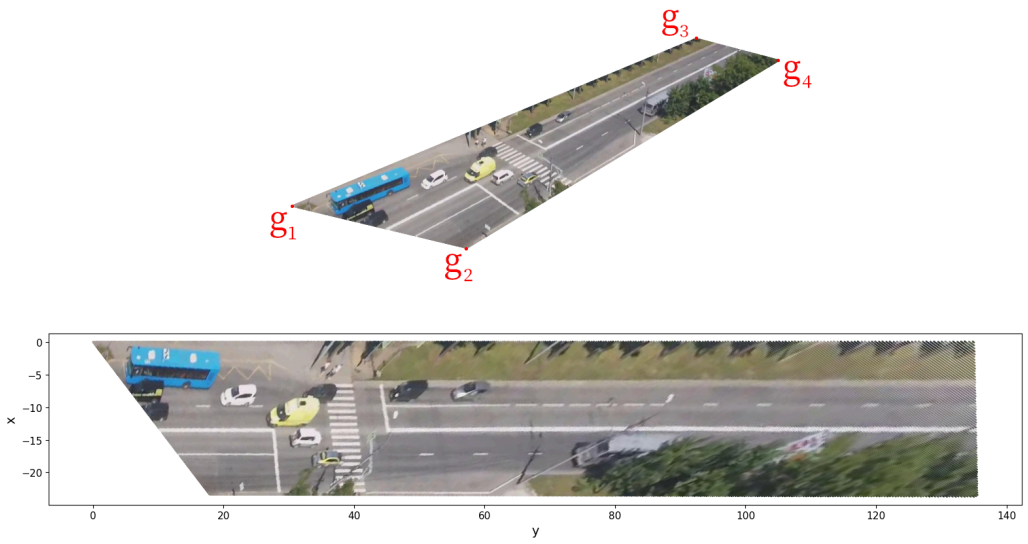

Figure 12. The Q_π set visualization in plane Π for case (50).

The distances in Q correspond to estimates obtained online and a simple rangefinder (accuracy up to a meter). We plan more accurate assessments using the geodetic tools.

We repeat the needed steps for hypothesis (51) and compare the results (see Figures 12 and 13).

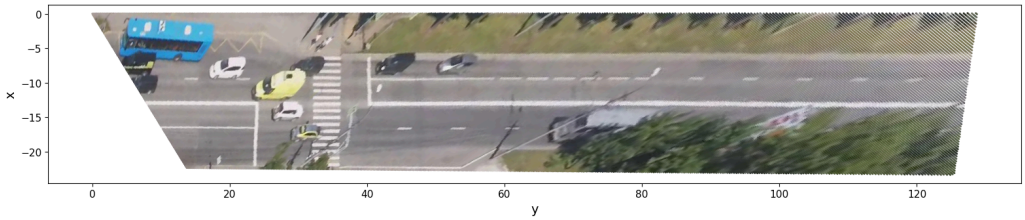

Figure 13. The Q_π set visualization in plane Π for case (51).

We observed a change in the geometry of Q_π for the hypothesis (51). For example, the pedestrian crossing changed its inclination; the road began to expand to the right. In this example, it is not easy to obtain several long parallel lines due to vegetation, and it is an argument for examining the cameras in a suitable season.

6. Conclusions

This work is an extension of the technology of road traffic data collection described in [4] to allow accurate car density and speed measurements using physical units with compensation for perspective and radial distortion.

Although we processed video frames in the article, all the mappings obtained for measurements $(\Theta, \xi, [R|t], R_{c2\pi})$ transform only coordinates. We used the pixel content of the images only for demonstrations. In the context of measurements, object detection (or instance segmentation) and object tracking algorithms work with pixel colors. The mappings work with the pixel coordinates of the results of these algorithms. We can calculate the discrete maps once and use them until the camera parameters or the Q area change. In this sense, the computational complexity of mapping generation is not particularly important. We plan to refine estimates of camera parameters as new data on the actual geometry of the ROI area will be available, and the accuracy of the maps will increase. Evaluating or refining the camera parameters in a suitable season might be better.

Author Contributions: Investigation, A.Z. and E.N.; Software, A.Z.; Visualization, A.Z.; Writing—original draft, A.Z.; Writing—review & editing, A.Z. and E.N. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the project 075-02-2022-880 of Ministry for Science and Higher Education of the Russian Federation from 31 January 2022.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Xinqiang, C.; Shubo, W.; Chaojian, S.; Yanguo, H.; Yongsheng, Y.; Ruimin, K.; Jiansen, Z. Sensing Data Supported Traffic Flow Prediction via Denoising Schemes and ANN: A Comparison. *IEEE Sens. J.* **2020**, *20*, 14317–14328.
2. Xiaohan, L.; Xiaobo, Q.; Xiaolei, M. Improving Flex-route Transit Services with Modular Autonomous Vehicles. *Transp. Res. Part E Logist. Transp. Rev.* **2021**, *149*, 102331.
3. 75 Years of the Fundamental Diagram for Traffic Flow Theory: Greenshields Symposium TRB Transportation Research Electronic Circular E-C149. 2011. Available online: http://onlinepubs.trb.org/onlinepubs/circulars/ec149.pdf (accessed on 17 September 2022).
4. Zatserkovnyy, A.; Nurminski, E. Neural network analysis of transportation flows of urban agglomeration using the data from public video cameras. *Math. Model. Numer. Simul.* **2021**, *13*, 305–318.
5. Camera Calibration and 3D Reconstruction. OpenCV Documentation Main Modules. Available online: https://docs.opencv.org/4.5.5/d9/d0c/group__calib3d.html (accessed on 18 March 2022).
6. Szeliski, R. *Computer Vision: Algorithms and Applications*, 2nd ed.; Springer: Cham, Switzerland, 2022.
7. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University Press: New York, NY, USA, 2003.

8. Transformations between ECEF and ENU Coordinates. ESA Naupedia. Available online: https://gssc.esa.int/navipedia/index.php/Transformations_between_ECEF_and_ENU_coordinates (accessed on 18 March 2022).
9. Forsyth, D.A.; Ponce, J. *Computer Vision: A Modern Approach*, 2nd ed.; Pearson: Hoboken, NJ, USA, 2012.
10. Peng, S.; Sturm, P. Calibration Wizard: A Guidance System for Camera Calibration Based on Modeling Geometric and Corner Uncertainty. *arXiv* **2019**, arXiv:1811.03264.
11. Camera Calibration. OpenCV tutorials, Python. Available online: http://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html (accessed on 18 March 2022).
12. Marchand, E.; Uchiyama, H.; Spindler, F. Pose estimation for augmented reality: A hands-on survey. *IEEE Trans. Vis. Comput. Graph.* **2016**, *22*, 2633–2651. [CrossRef] [PubMed]
13. Perspective-n-Point (PnP) Pose Computation. OpenCV documentation. Available online: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html (accessed on 18 March 2022).
14. Liu, C.-M.; Juang, J.-C. Estimation of Lane-Level Traffic Flow Using a Deep Learning Technique. *Appl. Sci.* **2021**, *11*, 5619. [CrossRef]
15. Khazukov, K.; Shepelev, V.; Karpeta, T.; Shabiev, S.; Slobodin, I.; Charbadze, I.; Alferova, I. Real-time monitoring of traffic parameters. *J. Big Data* **2020**, *7*, 84. [CrossRef]
16. Zhengxia, Z.; Zhenwei, S. Object Detection in 20 Years: A Survey. *arXiv* **2019**, arXiv:1905.05055v2.
17. Huchuan, L.; Dong, W. *Online Visual Tracking*; Springer: Singapore, 2019.
18. Yandex Maps. Available online: https://maps.yandex.ru (accessed on 18 March 2022).