**Birdview. Detailed description**

For reference: further in the text the expression ‘corner of the chessboard’ (this element is highlighted in red in the figure below and is located at the intersection of black and white squares) is replaced by the expression ‘node of the chessboard’ for better understanding.

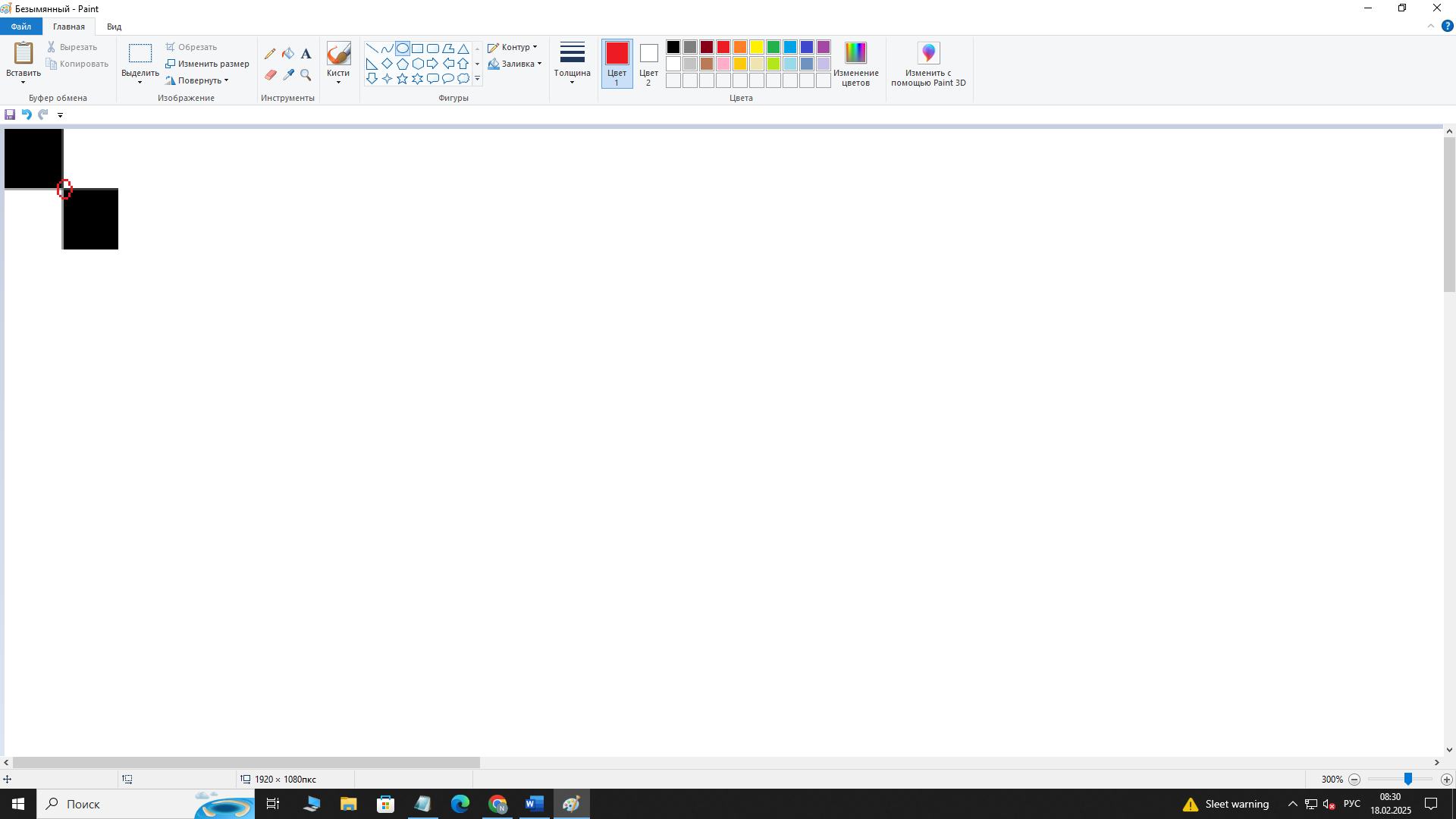


Figure 1. A corner (node) of a chessboard.

Step 1: Initialise the camera.

The VideoCapture() class from the OpenCV(cv2) library was used to initialise the camera. With the help of this class it is possible to work with different cameras as well as with video files. The class allows you to easily configure camera parameters (resolution, frame rate, brightness and contrast) using various methods.

Step 2: Camera Calibration.

Camera calibration is the task of obtaining the internal and external parameters of a camera from existing photos or videos captured by it. The checkerboard is one of the most popular objects for camera calibration for several reasons:

1. Clear angles and contrasting borders:

A checkerboard consists of alternating black and white squares, which creates clear contrasting boundaries. These boundaries are easily detected by computer vision algorithms, allowing for accurate corner coordinates.

2. Distortion Resistance:

The chessboard can be used in different lighting conditions and viewing angles. Due to its clear boundaries and known geometry, it remains a reliable object for calibration even in the presence of some distortion or noise in the image.

3. Known geometry:

The dimensions of the chessboard and the location of the squares are known in advance, so it is known exactly where the corners are on the real surface. This allows these known coordinates to be used as a reference for calibration.

4. Usability.

According to the programme code, the criterion for the accuracy of finding the corners of the chessboard is first established: the optimal number of iterations (30 iterations) with an accuracy of 0.001 is performed in terms of time and final quality. Then, after determining the size of the chessboard by nodes and the size of the output image, an array arr is created to store the coordinates of the chessboard nodes: the first two columns (arr[:, 0:2]) contain the initial coordinates of the chessboard corners in the image (in pixels), and the next two columns (arr[:, 2:4]) contain the refined coordinates of these corners obtained by subsampling.

The loop then captures frames from the camera, converts them to greyscale, and searches for chessboard nodes. Grayscale conversion is necessary because colour images contain three channels (red, green and blue) which makes them more complex to process and grayscale conversion reduces the amount of information to a single channel, hence due to the increased robustness to colour changes and reduced colour noise, node detection is improved. If chessboard nodes are found, they are refined using the cornerSubPix function and highlighted on the screen.

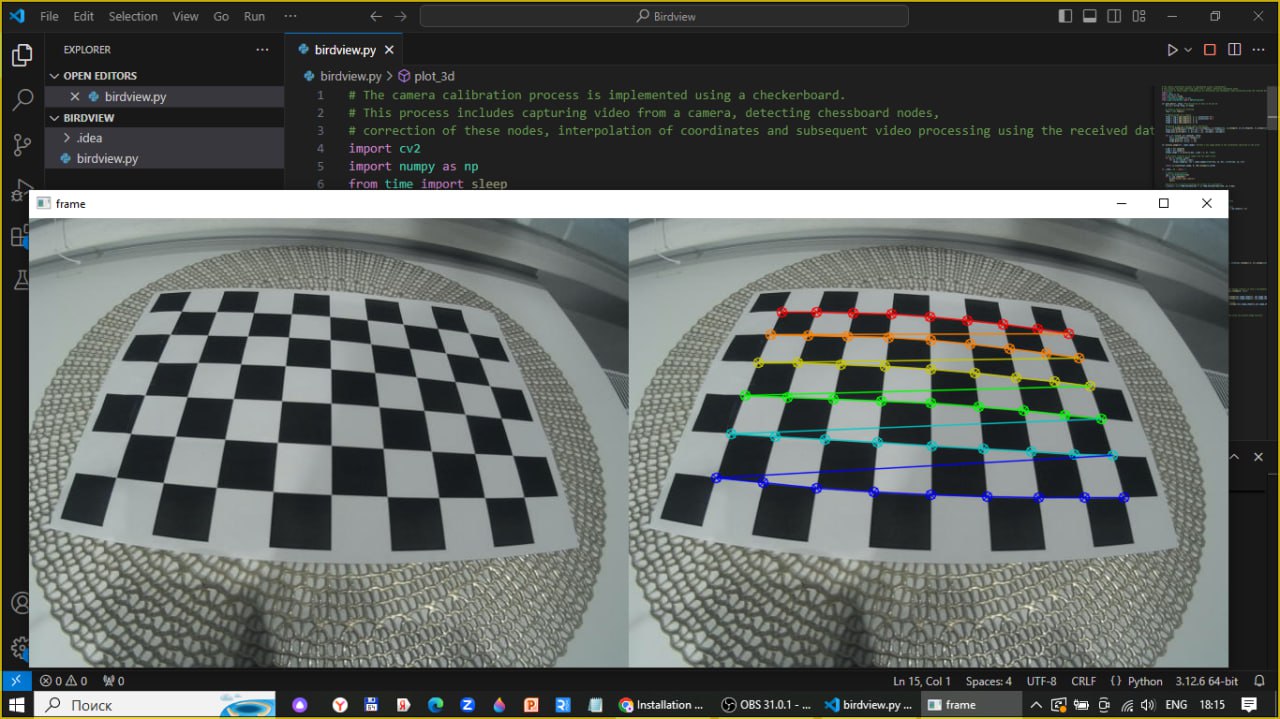


Figure 2. Checkerboard node detection (right) and normal video (left).

Step 3. Interpolation.

Interpolation is a method of finding intermediate values of a quantity from an existing discrete set of known values. In video processing, using coordinate interpolation techniques to create smoother transitions between pixels may be preferable for the following reasons:

1. Motion smoothing:

Coordinate interpolation can more accurately model the motion of objects in the frame, which is especially important when creating smooth transitions. This helps to avoid abrupt changes that can occur when simply changing pixels.

2. Improved image quality:

Coordinate interpolation can more effectively distribute colour values between pixels, resulting in more natural gradients and reducing the visibility of artefacts such as ‘jaggies’ or ‘steps’.

3. Preservation of detail:

Interpolation techniques can better preserve image detail, especially when zooming in or zooming out. This is especially important for videos where details can be lost by simple stretching or compressing, such as in the case of chessboard node detection.

After calibration, the RBFInterpolator function is used twice for coordinate interpolation: a smooth function is created that describes how to move pixels based on the calibration data. RBFInterpolator performs radial-basis interpolation. This is a method that interpolates data in a multidimensional space. In general, radial-basis interpolation (RBF interpolation) is an interpolation method that uses radial-basis functions to construct a smooth surface that passes through specified points. Radial basis functions are functions that depend only on the distance to some point (centre). Conceptually, radial basis functions resemble placing a rubber membrane over measured reference points and simultaneously reducing the overall curvature of the surface. The choice of basis function determines how the rubber membrane will be placed between the values. The figure below (Figure 3) conceptually shows the application of a radial basis function surface to a series of elevation reference points. In this case, the surface in cross section passes through the data values.

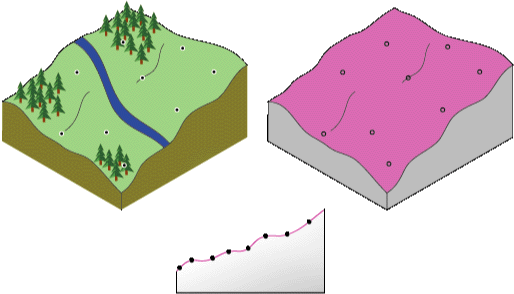


Figure 3. Illustration of how the surface in cross section passes through the data values.

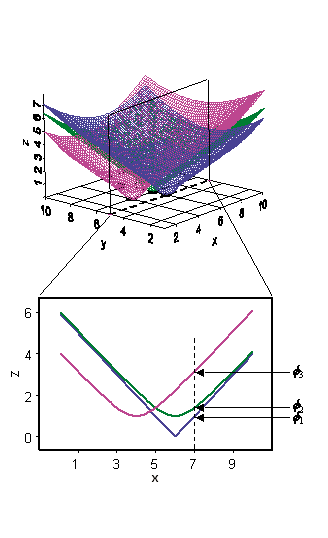


Figure 4. Arbitrary example of applying radial basis functions for different locations.

The advantages of RBF interpolation are smoothness (RBF interpolation provides smooth surfaces and is well suited for tasks requiring smooth transitions between values), flexibility (the method can handle arbitrary multidimensional data and does not require a regular grid), and locality (radial basis functions can be local, allowing efficient processing of data). The disadvantages of this interpolation include computational complexity, which leads to increased latency of the transformed video image, incorrect results when extrapolating (calculating values beyond the original data), due to which extrapolation methods are not used in this programme.

The results of the interpolation are used to create a pixel movement array, which stores information about how each pixel in the original image should be moved to create the new image. Each element of the array contains a pair of coordinates (y, x) indicating where the corresponding pixel in the original image should be moved.

At the end of this step, the interpolated pixel coordinates are visualised. The plot\_3d function creates 3D plots for two datasets (e.g., x and y coordinates) and 2D plots for their projections, initially using the plot\_wireframe method to build a grid.

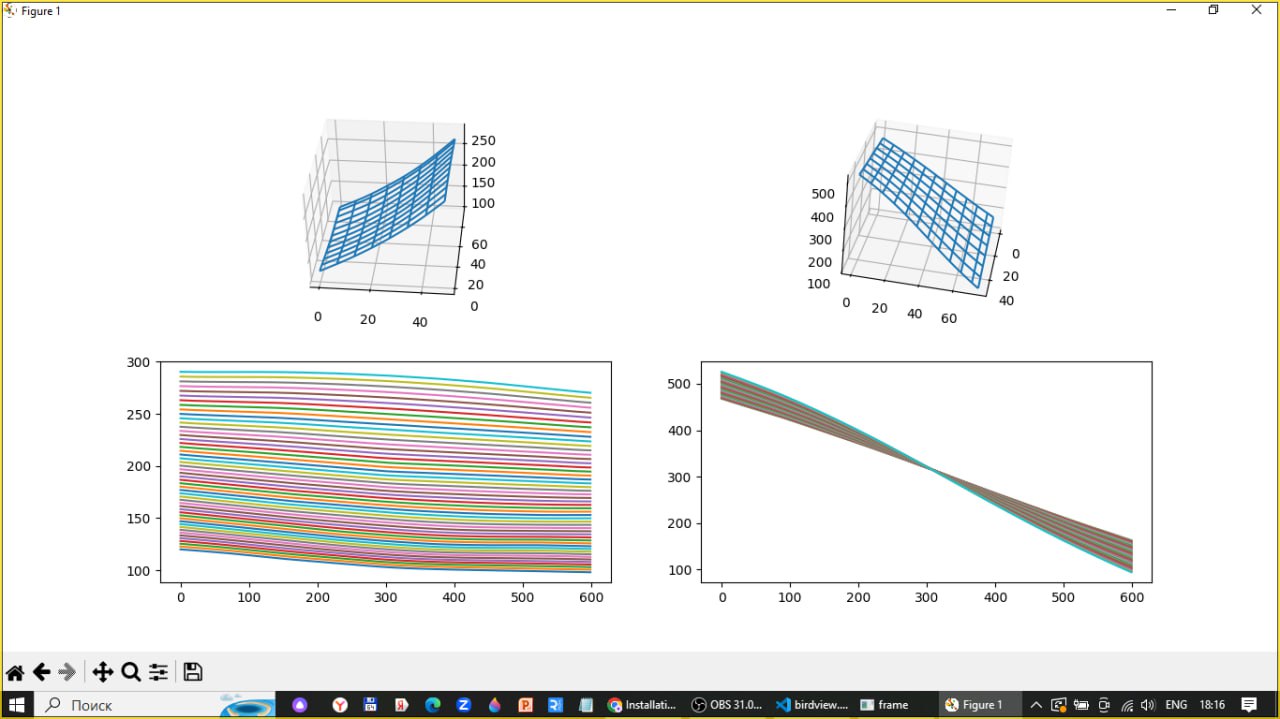


Figure 5. Interpolation results in graphs.

Step 4: Video Conversion.

In the last step in the loop, each frame is processed using the process\_image function to create a new image given the interpolated coordinates. For this purpose, each frame is transformed by the process\_image function based on the previously obtained calibration data.

The process\_image function:

Initially, an empty output\_image is created with one pixel extra width (to store RGB values). Then two nested loops go through each pixel within the array size and for each pixel in the output\_image the values from input\_image are assigned using the coordinates specified in the arr array.

Link to a youtube video with a detailed demonstration of the results of the programme: <https://www.youtube.com/watch?v=S8Jwaaax_qQ>

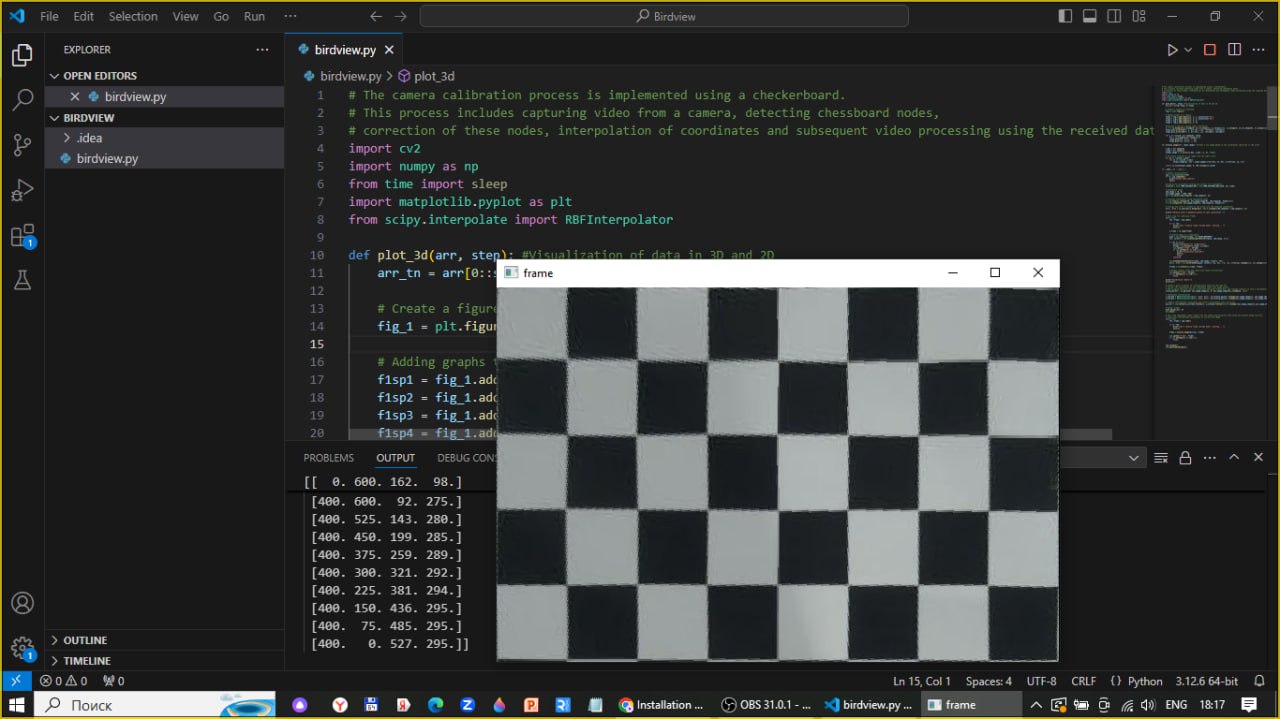


Figure 6. Pixel transfer table and transformed video image of the chessboard.

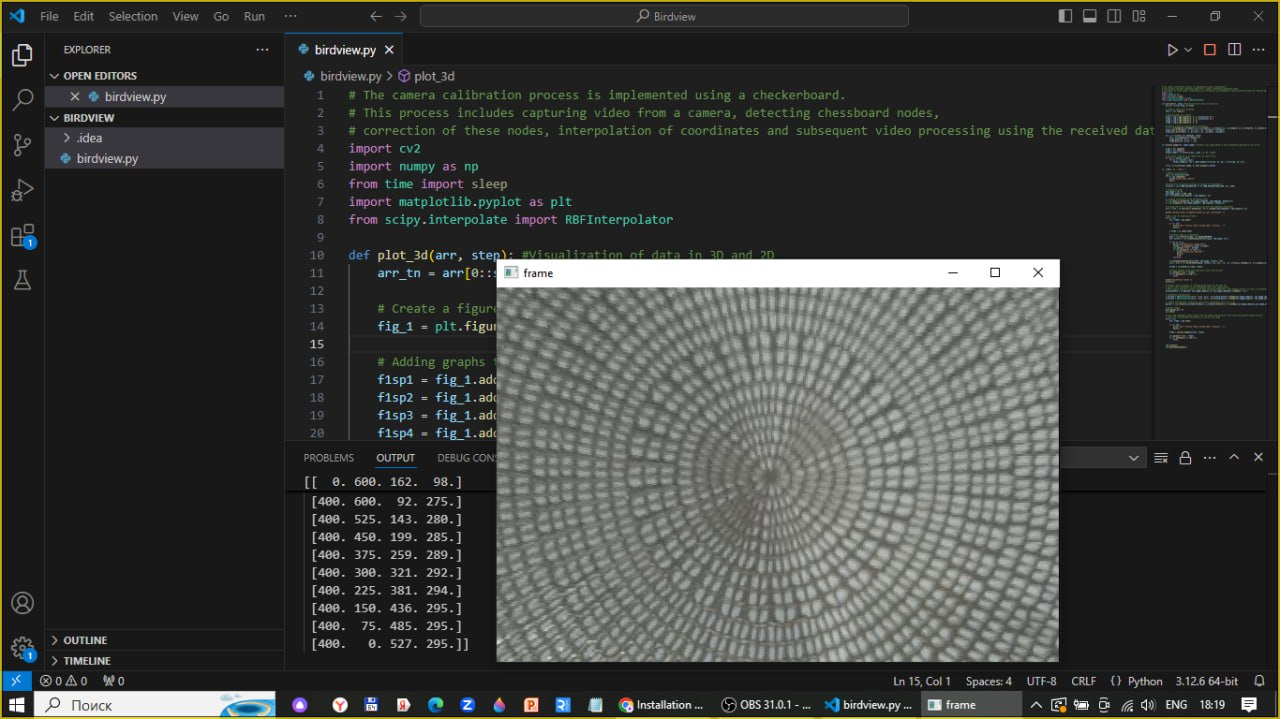


Figure 7. Unmodified pixel transfer table and transformed video image of the mat under the chessboard, which can be seen in Figure 2.