DLINE JOURNALS

Journal of Multimedia Processing and Technologies

Print ISSN: 0976-4127
Online ISSN: 0976-4135

JMPT 2024; 15 (2)
https://doi.org/10.6025/jmpt/2024/15/2/45-57

Influence of Neural Networks on the precision of image Processing

Saida S. Beknazarova
Tashkent University of Information Technologies named by Muhammad Al-Khwarizmi, A.Timur, 108, Tashkent, 100096
Uzbekistan
saida.beknazarova@gmail.com(S.S.Beknazarova)
0000-0001-7708-7616 (S.S. Beknazarova)

Dinara K. Kozhamzharova
International Information Technology University
Manas St. 34/1, Almaty, 050040, Kazakhstan
d.kozhamzharova@iitu.edu.kz (D.K. Kozhamzharova)
0000-0002-4320-9774 (D.K. Kozhamzharova)

Received: 18 January 2024
Revised: 29 March 2024
Accepted: 8 April 2024

ABSTRACT

This paper delves into a novel area of research, exploring how natural events impact the functioning of visualization systems, particularly in maintaining the integrity of the image processing algorithm. As part of this study, a unique filter was devised to eliminate distortions caused by fog. A correction filter was innovatively developed, and an in-depth analysis of the neural network's performance with images of varying resolutions was undertaken, leading to insightful recommendations for augmenting the precision of image processing.

Keywords: Image Processing, Image Filtering, Visualization Systems, Specific Identification

1. Introduction

The degree of influence of natural phenomena on images in visualization systems can vary depending on the type of natural phenomena and the specific characteristics of the visualization system. Here are some key considerations:

Weather Conditions: Outdoor Scenes: In outdoor visualization, weather conditions such as sunlight, clouds, rain, fog, and snow can significantly impact the appearance of images. These factors influence lighting conditions, visibility, and overall image clarity.

Indoor Scenes: While indoor scenes are less directly affected by weather, changes in natural light through windows can still impact the visual perception

within the visualization.

Time of Day: The position of the sun in the sky at different times of the day can create varying lighting conditions. This can influence the shadows, highlights, and color temperature of the scene,affecting the realism of the visualization.

Seasonal Changes: Natural phenomena associated with different seasons, such as changes in foliage, snow cover, or the angle of the sun, can have a significant impact on the visual appearance of outdoor scenes in visualization systems.

Geographic Location: The geographic location of a scene can affect the angle and intensity of sunlight, as well as other environmental factors. For example, scenes near the equator may experience more direct sunlight, while those at higher latitudes may have longer shadows.

Dynamic henomena: Natural dynamic phenomena, such as flowing water, moving clouds, or vegetation swayin in the wind, can add realism to visualizations. Simulating these dynamic elements accurately requires sophisticated algorithms and computational resources.

Simulation Accuracy: The degree to which natural phenomena are accurately simulated in a visualization system will impact the realism of the images. Advanced systems may incorporate physics-based models to simulate the behavior of light, water, and other elements realistically.

Adaptive Systems: Some visualization systems are designed to adapt to changing natural conditions in real time. For example, systems used in flight simulations might adjust lighting and visibility parameters based on the time of day and current weather conditions.

User Control: Depending on the application, users may have control over certain aspects of natural phenomena in the visualization. For instance, users might be able to adjust the time of day or toggle between different weather conditions.

Understanding and managing the influence of natural phenomena in visualization systems is crucial for creating realistic and immersive virtual environments, whether for entertainment, training simulations, or scientific research. Advances in computer graphics and simulation technologies continue to improve the fidelity of these systems, allowing for a more accurate representation of the impact of natural elements on visualizations.

Video analytics, also known as video content analysis (VCA), is the use of computer algorithms and machine learning techniques to extract useful information from video footage. It involves analyzing video data in real-time or post-processing to automatically identify and track objects, detect events or anomalies, and extract meaningful insights from the video content.

Video analytics can be applied in various domains and industries, including security and surveillance, retail, transportation, healthcare, and more.

Here are some common applications of video analytics:

1. Object Detection and Tracking: Video analytics can identify and track objects of interest, such as people, vehicles, or specific objects, throughout the video. This can be used for security purposes, crowd monitoring, or counting the number of objects in a scene.

2. Activity Recognition: Video analytics can analyze human behavior and recognize specific activities or actions in the video, such as walking, running, loitering, or fighting. This can be useful in security and public safety applications.

3. Facial Recognition: Video analytics can detect and recognize faces in video footage, enabling applications like access control, identity verification, or tracking individuals of interest.

4. Crowd Analysis: Video analytics can analyze the behavior and movement patterns of a crowd in real time. It can detect crowd density, and crowd flow direction, or identify unusual

crowd behavior, which can be useful for managing public spaces and events.

5. Anomaly Detection: Video analytics can detect and alert on unusual or suspicious activities in a video, such as abandoned objects, intrusion, or unauthorized access.

6. Traffic Monitoring: Video analytics can analyze live video feeds from traffic cameras to monitor traffic flow, detect congestion, or identify traffic violations.

7. Retail Analytics: Video analytics can be used in retail stores to analyze customer behavior, measure footfall, track product placement, or identify patterns for optimization of store layouts and marketing strategies.

Video analytics has the potential to save time, improve efficiency, enhance security, and provide valuable insights in various applications. However, it is important to consider privacy concerns and ensure appropriate data protection and consent while implementing video analytics systems. Currently, the use of video surveillance systems has been introduced in many areas. Problems such as use, detection, object tracking, and segmentation occur in video sequence analysis. Weather conditions play a large role when it comes to outdoor video surveillance systems, which can significantly reduce image quality.

Image preprocessing refers to a set of techniques and operations performed on an image prior to its analysis or further processing. The goal of image preprocessing is to enhance the quality and usability of the image data, making it more suitable for specific tasks such as image recognition, object detection, or image segmentation.

Here are some common image preprocessing techniques:

1. Image resizing: Resizing an image involves changing its dimensions, either scaling it up or down. This can be done to standardize the size of images in a dataset or to fit the image into a specific resolution requirement for a particular application.

2. Image cropping: Cropping an image involves removing unnecessary or irrelevant parts of the image, focusing only on the region of interest. This can eliminate distractions and reduce the computational load for subsequent analysis.

3. Image normalization: Normalizing an image involves adjusting the pixel values to a standardized range or distribution. This can include operations such as scaling pixel values to a specific range, converting to grayscale, or applying adaptive histogram equalization to enhance contrast.

4. Image denoising: Denoising techniques are used to reduce noise or artifacts present in the image. This can involve methods like median filtering, Gaussian smoothing, or bilateral filtering to remove unwanted noise while preserving important details.

5. Image enhancement: Enhancement techniques aim to improve the visual quality or highlight specific features in an image. This can include operations such as sharpening, contrast adjustment, or local histogram equalization.

6. Image rotation and flipping: Rotating an image involves rotating it by a specified angle while flipping can include operations like flipping horizontally or vertically. These operations can be useful for data augmentation, aligning images, or correcting orientation.

7. Color space conversion: Converting an image from one color space to another can provide a different representation of the image, emphasizing different color characteristics. Common conversions include RGB to grayscale, RGB to HSV, or RGB to LAB color space.

8. Image filtering: Filtering operations, such as edge detection or blurring, can be applied to enhance or extract specific features in an image. These operations can help in subsequent analysis or feature extraction.

9. Image thresholding: Thresholding techniques are used to convert an image into a binary format, where pixels are classified as foreground or background based on their intensity values. This can be useful for segmentation tasks or extracting specific regions of interest.

These are some examples of image preprocessing techniques, and the choice of techniques depends on the specific requirements and goals of the image analysis task. Preprocessing can significantly impact the accuracy and effectiveness of subsequent image analysis algorithms, so it's important to consider and apply the appropriate techniques based on the specific application.

In technical vision systems, image preprocessing plays a crucial role in preparing images for analysis and processing. Some specific image preprocessing techniques commonly used in technical vision systems include:

1. Image normalization: Normalization techniques are used to standardize the pixel values of an image, ensuring that they fall within a specific range or distribution. This can be done to compensate for differences in illumination conditions or to facilitate comparison between images.

2. Image denoising: Denoising methods are used to reduce noise or artifacts present in the image. This can include techniques such as median filtering, Gaussian smoothing, or wavelet denoising, which help to improve the signal-to-noise ratio or eliminate unwanted noise.

3. Image registration: Image registration techniques are used to align multiple images captured from different viewpoints or at different times. This can involve methods like feature-based registration, intensity-based registration, or geometric transformations to correct for any misalignments or distortions.

4. Image segmentation: Image segmentation is the process of partitioning an image into different regions or objects of interest. Preprocessing techniques, such as edge detection, thresholding, or region growing, can be applied to enhance the quality of the segmentation results by isolating and separating the desired objects.

5. Image enhancement: Enhancement techniques aim to improve the visual quality or highlight specific features in an image. In technical vision systems, enhancement methods like contrast stretching, histogram equalization, or spatial filtering may be used to enhance relevant details or improve the visibility of certain image features.

6. Image feature extraction: Prior to object recognition or classification, it is often necessary to extract relevant features from an image. Preprocessing techniques such as edge detection, corner detection, or scale-space analysis can be used to identify and extract important visual features that can be used for subsequent analysis.

7. Image resizing: Resizing an image can be performed to standardize the size of images in a dataset or to adapt images to a specific processing requirement. This may involve scaling up or down the image dimensions while maintaining the aspect ratio.

8. Color space conversion: Converting an image from one color space to another can provide a different representation of the image, with potential benefits for specific tasks. For example, converting an RGB image to a grayscale or HSV color space can facilitate certain types of analysis or feature extraction.

Overall, image preprocessing in technical vision systems involves a combination of techniques to enhance image quality, reduce noise, align images, segment objects, enhance features, and extract relevant information. These preprocessing steps are typically performed before the image data is fed into algorithms for further analysis and processing.

One of the atmospheric phenomena is fog, which can significantly reduce the quality of image

recognition and recognition. Images taken in foggy weather will be low-saturated and low contrast.

Due to the presence of fog and steam, light diffuses into the atmosphere before reaching the chamber. Fog is known to affect the perception of the distance between the camera and the object. Therefore, fog loss requires an assessment of depth or light scattering. If the input is only one ambiguous image, depth estimation is a minor problem. Stereoscopic depth estimation requires two images. Therefore, many have proposed methods that use several of the proposed images Schechner et al-suggesting a polarization-based method.

This method removes fog using two or more images taken at different polarization levels. Fattal proposed a method based on independent component analysis (independent component analysis). This method assesses optical conductivity in inaccurate images. Based on this assessment, it proposed a method to reduce broken light to improve visibility and eliminate fog. This method assesses optical conductivity in inaccurate images. Based on this assessment, broken light is eliminated to improve visibility and eliminate fog. Recovery is based on color data, so this method is not applicable to a gray image. In addition, this method does not work even when there is dark fog, since dark fog is often colorless. Tan proposed a method based on spatial ordering over an image on a single-color or grayscale. The work eliminates fog by increasing local image contrast, but the reconstructed image appears to be very saturated. There are also methods based on the use of a 3D scene Model, graph-based image segmentation, etc [4]. This article proposes a fog image filtering algorithm to improve the accuracy of object detection.

2. Methodology

The degree to which natural phenomena are accurately simulated in a visualization system is commonly referred to as "fidelity" or "accuracy." Fidelity in this context relates to how closely the simulation or representation in the visualization mirrors the actual behavior of the realworld phenomena it aims to depict.

Several factors contribute to the fidelity of a visualization system:
Mathematical Accuracy: The underlying mathematical models and algorithms used to simulate natural phenomena must be accurate and reliable. This involves understanding the physics, chemistry, or other relevant principles governing the phenomenon.

Data Quality: The accuracy of the data used in the simulation is crucial. If the input data is imprecise or outdated, it can lead to inaccuracies in the visualization.

Resolution: The resolution of the simulation, both spatial and temporal, plays a significant role. Higher resolutions allow for more detailed and accurate representations of phenomena.

Rendering Quality: The visual representation of the simulation also contributes to fidelity. High-quality rendering techniques can enhance the realism of the visualization.

Real-time Interaction: For interactive visualization systems, the ability to provide real-time feedback and allow users to interact with the simulation can enhance the overall fidelity.

Sensory Fidelity: In some cases, especially in virtual reality or immersive simulations, the fidelity may also refer to how well the visualization engages multiple senses, such as sight, sound, and touch, to create a more realistic experience.

Achieving high fidelity is essential in various fields, including scientific research, engineering design, and virtual training simulations. The more accurately a visualization system can replicate natural phenomena, the more valuable it becomes for analysis, decision-making, and understanding complex processes.

Fog as an image distortion factor
Fog is the accumulation of condensation products on the surface of the Earth and, as a result, a strong clouding of air, horizontal vision will be less than 1 km. At positive air

temperatures, the fog consists of small drops of water, around 10°C, and at low temperatures, fog crystals appear, that is, mixed. In this case, only a district of drops is considered. Now it is necessary to explain how vision decreases in the presence of fog. As mentioned above, the fog consists of drops of water, and when light passes through each drop, it spreads, resulting in a hoarse image. Therefore, for processing images, it is necessary to use a filter that eliminates the scattering caused by fog. It is known that the elastic scattering of radiation occurs in the fog. Elastic scattering is characterized by the fact that it is not accompanied by internal changes. The states of the scattering particle and its effect on it are studied, which is especially observed in fog. The most typical types of elastic scattering are: Tindall scattering (effect), Reyl scattering, Mie scattering. The Tindall effect occurs when light radiation is propagated by suspended particles that are small in relation to the size of wavelengths. The distribution of the Tindal is elastic, i.e. when a photon strikes a particle it is absorbed and re-emitted in any direction, and the dimensions of the suspended particles are many times larger than the dimensions of the atoms. This is typical for colloidal systems, such as smoke, fog, gels, etc. An example of a Tindall effect is blue eye color. It is known that a small amount of melanin is observed in blue eyes, as a result of which the red part of the spectrum does not spread and is absorbed by melanin when passing through the cornea, while the blue part of the spectrum is spread by melanin molecules and collagen fibers, thanks to which blue color can be observed. Reyl scattering is diffractive and therefore manifests itself in particles smaller in size than the wavelength. In the process of propagation, the frequency of light does not change [7]. It is important to note that the propagation intensity increases with a decrease in wavelength: the shorter the wave, the stronger the propagation. The most common example of relay scattering is the sky color (a mixture of different gases) produced when waves of different lengths and intensities (sunlight) pass through the atmosphere.

Mie scattering is observed when the radiation wavelength is larger than the scattering particle size [8]. This type of scattering is characteristic of fog and, moreover, most accurately describes the process of deflection of light rays when passing through fog drops. Such accuracy is achieved by representing scattering in the form of gradually converging rows, but the practical use of such a row is much more problematic, so it is usually expressed in the form below:

$$\chi(\gamma) = \frac{1-g^2}{(1+g^2-2g\cos\gamma)^{\frac{3}{2}}},$$

Where $\chi(\gamma) - \gamma$ is the proportion of reflected light in the direction of the γ is the angle, where $x_{i,j}$, g is the anisotropy factor. The anisotropy factor is determined by the size of the scattering particles, the wavelength, and the distribution of the particle size.

To calculate the effect of scattering and the visible angle of the Pixel, we use the following symbol. When blurring, the Pixel is changed according to a certain rule, so instead of one pixel, many pixels of different intensity are formed.

The red color in the image indicates the pixel of the highest intensity, followed by pixels in the order of decreasing intensity, conditionally highlighted in green, yellow, and blue. The Centers of pixels are represented by symbols, the distance of two adjacent centers. Conditionally, the pixel size is 1, and the Pixel 1 is 1 square, respectively, y = 1. The angle at which the Pixel center is located is defined γ in the optical center of the system. Focal length (d) 8 relative units were adopted as part of the study.

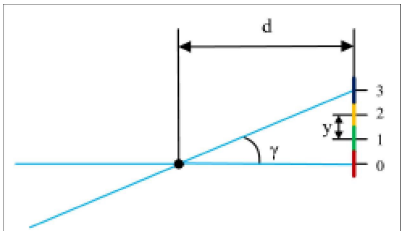

Figure 1. Dotted image formation scheme

So now you need to calculate the angles γ as follows: $\gamma = arctg(\frac{y}{d})$. Then the intensity for each angle is calculated using the formula. Thus, we get the following Pixel distribution (circles define calculated intensity fields):

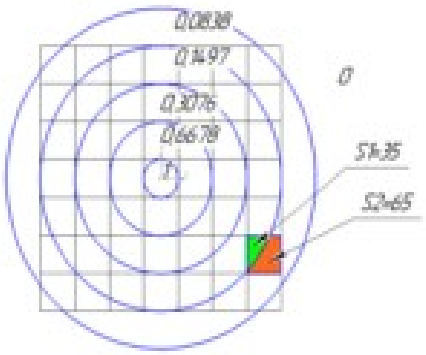

Figure 2. Location of pixels in intensity fields

Now you can calculate the radiation energy per pixel. To do this, we calculate the intensity that falls on the Pixel. The Pixel is divided into two regions, which are included in zones of different intensity. The average weight is then found:

$$I_{\text{ср}} = \frac{s_1 * I_1 + s_2 * I_2}{s_1 + s_2}.$$

3. Results and Discussion

Adaptive systems, particularly in the context of visualization, refer to systems that can dynamically adjust their presentation or behavior in response to changing conditions or input data. In the case of visualization systems designed to adapt to changing natural conditions in real-time, several key characteristics and features may be involved:

Real-time Data Integration: These systems are equipped to handle and integrate real-time data from various sources. For example, a weather visualization system might integrate live weather data from sensors, satellites, or other sources to provide up-to-the-minute information.

Dynamic Parameter Adjustment: The visualization parameters, such as colors, scales, and data ranges, may be dynamically adjusted based on the changing natural conditions. This ensures that the visualization remains relevant and effective as conditions evolve.

Responsive Design: The system may have a responsive design that allows it to adapt to different screen sizes or resolutions. This ensures that the visualization remains clear and readable across various devices.

Machine Learning Algorithms: Machine learning algorithms can be employed to analyze patterns in changing conditions and automatically adjust the visualization parameters. For instance, an adaptive traffic visualization system might use machine learning to predict congestion patterns and adjust the display accordingly.

User Interaction: Adaptive systems may allow user interaction to influence the visualization. Users might be able to customize certain aspects or provide feedback that the system uses to adapt its presentation.

Multi-modal Data Integration: Integration of data from multiple modalities, such as combining visual and auditory cues, can enhance the adaptability of the system. For example, a disaster visualization system might incorporate not only visual information but also audible alerts.

Scalability: The system should be scalable to handle varying levels of data complexity and volume. This ensures that it remains responsive and efficient even as the scale of the data changes.

Feedback Mechanisms: Feedback loops can be incorporated to continuously evaluate the effectiveness of the visualization in conveying information. Based on feedback, the system can autonomously or semi-autonomously adjust its parameters.

Predictive Analytics: Some adaptive systems may employ predictive analytics to anticipate future changes in natural conditions. This allows the system to proactively adapt before the changes occur.

Adaptive visualization systems find applications in various domains, including weather monitoring, traffic management, environmental monitoring, and more. Their ability to provide timely and relevant information in a dynamically changing environment makes them valuable tools in decision-making processes.

For best results, you should first apply the Mi blur filter, and then use it to synthesize the reverse filter to eliminate the spread.

The development of the filter is carried out in several stages:

1. To build an indicator and determine the minimum angle value, which must be taken into account when calculating the intensity for each pixel.

2. Determination of angles at which Pixel centers are visible from the optical center of the system.

3. Calculate the intensity for each pixel and construct a blur Matrix.

4. Testing a blurred Matrix.

5. Change the dim matrix to get the filter.

Thus, we carry out the above steps of filter synthesis. By calculating each element using the formula n, a series of intensity values is formed. For angles, the intensity was calculated 0-90 (Step-1) after which the values are normalized, that is, each element of the array is divided by a maximum. Next, it is necessary to build a scattering indicator to determine the maximum value of the angle that must be taken into account when calculating the intensity for each pixel. Note that angles are expressed in the calculation.

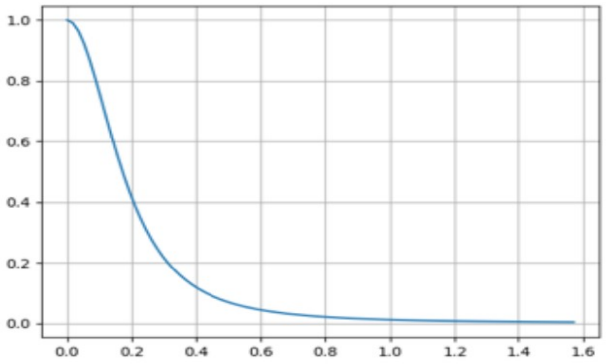

Figure 3. Scattering indicator to determine the maximum value of the angle

The graph shows that in subsequent calculations it is recommended to take into account angles with a value of less than 0.5 rad since as the values increase, the intensity becomes almost zero. The desired turbidity matrix is presented in Table 1.

The intensity of the central (original) Pixel is 1, that is, it does not change, but when blurring, its intensity must also be reduced, so it is necessary to normalize the matrix formed by dividing each element by the sum of all elements. The normalized opacity matrix is presented in Table 2.

0,066202	0,066202	0,066202	0,066202	0,066202	0,066202	0,066202
0,106865	0,106865	0,106865	0,106865	0,106865	0,106865	0,106865
0,139156	0,139156	0,139156	0,139156	0,139156	0,139156	0,139156
0,1497	0,1497	0,1497	0,1497	0,1497	0,1497	0,1497
0,139156	0,139156	0,139156	0,139156	0,139156	0,139156	0,139156
0,106865	0,106865	0,106865	0,106865	0,106865	0,106865	0,106865
0,066202	0,066202	0,066202	0,066202	0,066202	0,066202	0,066202

Table 1. Non-standard opacity Matrix

0,005254	0,005254	0,005254	0,005254	0,005254	0,005254	0,005254
0,008482	0,008482	0,008482	0,008482	0,008482	0,008482	0,008482
0,011045	0,011045	0,011045	0,011045	0,011045	0,011045	0,011045
0,011882	0,011882	0,011882	0,011882	0,011882	0,011882	0,011882
0,011045	0,011045	0,011045	0,011045	0,011045	0,011045	0,011045
0,008482	0,008482	0,008482	0,008482	0,008482	0,008482	0,008482
0,005254	0,005254	0,005254	0,005254	0,005254	0,005254	0,005254

Table 2. Normalized dimming Matrix

Before starting to change the blur matrix, it is necessary to take into account the increase in

-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254
-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482
-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045
-0,011882	-0,011882	-0,011882	-0,011882	-0,011882	-0,011882	-0,011882
-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045
-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482
-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254

Table 3. Normalized precision Matrix

image resolution due to the increase in contrast between the central Pixel and its neighbors. To achieve this goal, all coefficients in the new Matrix, except for the central element, must take negative values, and the value of the central element must be greater than its value in the blurred Matrix. Thus, the transformation of the central element is carried out by dividing it into one; the remaining coefficients are multiplied by -1. The unorthodox Clarity Matrix is presented in Table 3.

When testing a filter, a situation arises when the original image is turned completely white. This can happen when the maximum value for the RGB model exceeds the upper limit - (255, 255, 255) - White. In this case, it is necessary to multiply the central element of the matrix by the coefficient, which received a value of less than one. As a result of selection, the optimal value of the multiplier was found - 0,148. The simplified Clarity Matrix is presented in Table 4. Converting an image to grayscale is a common preprocessing step in computer vision applications. It involves transforming an image that is originally represented in color (RGB) into a grayscale image, where each pixel is represented by a single intensity value.

The conversion from color to grayscale is typically achieved by taking a weighted average of the RGB channels. There are various methods to perform this conversion, but one commonly used method is the luminosity method, which takes into account the human perception of color.

.

-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254
-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482
-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045
-0,011882	-0,011882	-0,011882	-0,011882	-0,011882	-0,011882	-0,011882
-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045	-0,011045
-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482	-0,008482
-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254	-0,005254

Table 4. Normalized precision Matrix

To convert an image to grayscale using the luminosity method, follow these steps
1. Read the image: Load the color image using a library or module that supports image processing (such as OpenCV or Pillow).

2. Convert to grayscale: Apply the following formula to each pixel in the image to calculate the grayscale intensity value: Grayscale Intensity = 0.21 * Red + 0.72 * Green + 0.07 * Blue Repeat this process for every pixel in the image, updating their RGB channels to the calculated grayscale intensity value.

3. Save the grayscale image: Save the modified image as a grayscale image format (such as JPEG or PNG) for further analysis or visualization.

Converting an image to binary is a common preprocessing step in computer vision applications, where we want to separate the foreground (objects of interest) from the background. In a binary image, each pixel is represented by only two possible values: 0 (black)
or 255 (white).

There are various methods to convert an image to binary, and the choice of method depends on the specific application and characteristics of the image. Here, I'll describe a commonly used method called thresholding, which is based on setting a threshold value to separate foreground and background pixels [11].

To convert an image to binary using thresholding, follow these steps:
1. Read the grayscale image: Load the grayscale image using a library or module that supports image processing (such as OpenCV or Pillow).

2. Choose a threshold value: Select a threshold value between 0 and 255, which will determine how intensity values are divided into foreground and background.

3. Apply thresholding: Iterate over each pixel in the grayscale image and compare its intensity value with the threshold value. If the intensity value is greater than or equal to the threshold, assign it a value of 255 (white). Otherwise, assign it a value of 0 (black).

4. Save the binary image: Save the modified image as a binary image format (such as JPEG or PNG) for further analysis or visualization [12].

Bernsen's method is one of the locally adaptive binarization methods developed for image segmentation. In this study, Bernsen's locally adaptive binarization method is implemented and then tested for different grayscale images.

The Bernsen method is a thresholding technique used for image binarization. It is based on local thresholding, where the threshold value for each pixel is determined based on the local neighborhood around it. The method aims to find a threshold that maximally distinguishes the foreground and background pixels.

Here is an overview of the Bernsen method for image binarization

1. Read the grayscale image: Load the grayscale image using a library or module that supports image processing (such as OpenCV or Pillow).

2. Define neighborhood size and contrast threshold: Specify the size of the local neighborhood and a contrast threshold. The neighborhood size determines the number of pixels considered around each pixel during thresholding. The contrast threshold is used to differentiate between foreground and background pixels based on their intensity variation [13].

3. Iterate over each pixel in the image: For each pixel, compute the local neighborhood intensity range (maximum-minimum intensity values).

4. Compute the threshold value: Apply the Bernsen formula to calculate the threshold value for each pixel based on the contrast threshold and the intensity range.

5. Binarize the image: Compare the intensity value of each pixel with its threshold value. Assign a value of 255 (white) if the intensity is greater than or equal to the threshold; otherwise, assign a value of 0 (black).

6. Save the binary image: Save the modified image as a binary image format (such as JPEG or PNG) for further analysis or visualization [14].

Here is an example code in Python using the OpenCV library to perform binarization using the Bernsen method

```
import cv2
import numpy as np
def bernsen_thresholding(image, neighborhood_size, contrast_threshold):
# Create an empty output image of the same shape as the input image
binary_image = np.zeros_like(image)
# Calculate the border size based on the neighborhood size
border_size = neighborhood_size // 2
# Iterate over each pixel in the image
for i in range(border_size, image.shape[0] - border_size):
for j in range(border_size, image.shape[1] - border_size):
# Extract the local neighborhood
local_neighborhood = image[i - border_size : i + border_size + 1, j - border_size : j + order_size + 1]
# Compute the intensity range within the neighborhood
```

```
    intensity_range = np.max(local_neighborhood) - np.min(local_neighborhood)
    # Compute the threshold value
    threshold = (np.max(local_neighborhood) + np.min(local_neighborhood)) / 2
    # Binarize the pixel based on the threshold value
    binary_image[i, j] = 255 if image[i, j] >= threshold - contrast_threshold / 2 else 0
    return binary_image
    # Read the grayscale image
    gray_image = cv2.imread('grayscale_image.jpg', cv2.IMREAD_GRAYSCALE)
    # Set the neighborhood size and contrast threshold
    neighborhood_size = 15
    contrast_threshold = 15
    # Apply Bernsen thresholding
    binary_image = bernsen_thresholding(gray_image, neighborhood_size, contrast_threshold)
    # Save the binary image
    cv2.imwrite('binary_image.jpg', binary_image)
```

Replace 'grayscale_image.jpg' with the filename and path of your input grayscale image and binary_image.jpg' with the desired output filename for the binary image.

In the above code, we defined the bernsen_thresholding function that takes the grayscale image, neighborhood size, and contrast threshold as input. This function computes the threshold value for each pixel using the Bernsen formula and applies the binarization based on the threshold value [15,16].

4. Conclusion

In conclusion, the degree of influence of natural phenomena on images in visualization systems is a multifaceted and dynamic aspect that requires careful consideration in various applications. Throughout this article, we have explored the impact of natural phenomena such as lighting conditions, atmospheric effects, and environmental variables on the quality and accuracy of images generated by visualization systems.

It is evident that these natural phenomena play a crucial role in shaping the visual output of simulation and rendering systems. The degree of influence varies across different scenarios, and understanding these factors is essential for creating realistic and immersive visualizations.

Advancements in technology, including the integration of sophisticated algorithms and the utilization of real-time data, have contributed to minimizing the negative impact of natural phenomena on visualization systems. However, challenges persist, especially in scenarios where unpredictable and extreme environmental conditions come into play.

Researchers and developers must continue to explore innovative solutions to enhance the robustness of visualization systems in the face of natural phenomena. This may involve the development of adaptive algorithms, improved sensors, and the incorporation of machine learning techniques to dynamically adjust rendering parameters based on changing environmental conditions.

Ultimately, the degree of influence of natural phenomena on images in visualization systems underscores the need for a holistic and interdisciplinary approach. Collaboration between experts in computer graphics, meteorology, environmental science, and related fields is essential to develop comprehensive solutions that address the challenges posed by varying natural conditions.

As technology continues to evolve, it is likely that visualization systems will become increasingly resilient to the influence of natural phenomena, providing users with more accurate and lifelike representations of virtual environments. This ongoing progress holds

promise for applications ranging from gaming and entertainment to scientific simulations and urban planning, where realistic visualizations are crucial for decision-making and user engagement.

Furthermore, atmospheric effects, such as haze, scattering, and reflection, contribute to the overall realism of the rendered images. These effects are crucial in accurately representing the interaction of light with the environment, ultimately enhancing the visual fidelity of the visualization system.

It is essential for visualization systems to accurately model and simulate these natural phenomena to create a compelling and immersive experience for users. By understanding the degree of influence of natural phenomena on images, developers can prioritize the implementation of realistic lighting, weather conditions, and atmospheric effects to achieve more convincing visualizations.

Overall, the degree of influence of natural phenomena on images in visualization systems cannot be underestimated. By carefully considering and incorporating these factors into the design and implementation of visualization systems, developers can create stunning and captivating visual experiences for users.

5. Acknowledgements

Image preprocessing is an important stage of image processing. The choice of image processing methods depends on the duration of the preliminary stage of image processing and the accuracy of the transmission of the characteristics of the initial objects of the working scene of an industrial robot.

The methods were evaluated on the basis of numerical experiments. The variable parameters included the size of the local area and empirically determined coefficients.

According to the results of the analysis, the most promising methods for the problem of binarization of images in robotics are the methods of Eikvil, Bernsen, and BPM. The methods demonstrated high-quality image processing of the working scene with uneven lighting, separation of the image into background and objects of interest, filtering out shadows from objects of interest.

References

[1] Schechner, Y. Y., Narasimhan, S. G., Nayar, S. K. (2001). Instant dehazing of images using polarization. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 325-332).

[2] Fattal, R. (2008). Single image dehazing. In *International Conference on Computer Graphics and Interactive Techniques archive ACM SIGGRAPH* (pp. 1-9). https://doi.org/10.1145/1399504.1360619.

[3] Tan, R. T. (2008). Visibility in bad weather from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1-8). https://doi.org/10.1109/CVPR.2008.4587643.

[4] Kopf, J., Neubert, B., Chen, B., Cohen, M., Cohen-Or, D., Deussen, O., Uyttendaele, M., & Lischinski, D. (2008). Deep photo: Model-based photograph enhancement and viewing. *ACM Transactions on Graphics*, 27(5), 116:1-116:10. https://doi.org/10.1145/1409060.1409076.

[5] Fang, S., Zhan, J., Cao, Y., Rao, R. (2010). Improved single image dehazing using segmentation. In *IEEE International Conference on Image Processing (ICIP)* (pp. 3589-3592). https://doi.org/10.1109/ICIP.2010.5653957.

[6] Matveyev, L. T. (1984). *Kurs obschei meteorologii. Fizika atmosfery*. Leningrad: Gidrometeodizdat.

[7] Beknazarova, S., Mukhamadiyev, A. Sh., Jaumitbayeva, M. K. (2019). Processing color images, brightness and color conversion. In *International Conference on Information Science and Communications Technologies (ICISCT 2019) Applications, Trends and Opportunities*. Tashkent.

[8] Beknazarova, S., Mukhamadiyev, A. Sh., Park, I., Adbullayev, S. (2020). The mask of objects in intellectual irrigation systems. In *International Conference on Information Science and Communications Technologies (ICISCT 2020) Applications, Trends and Opportunities*. Tashkent.

[9] Beknazarova, S., Sadullaeva, Sh., Abdurakhmanov, K., Beknazarov, K. (2020). Nonlinear cross-systems of numerical simulation of diffusion processes. In *International Conference on Information Science and Communications Technologies (ICISCT 2020) Applications, Trends and Opportunities*. Tashkent.

[10] Korikov, A. M., Syryamkin, V. I., Titov, V. S. (2000). *Correlation visual systems of robots* (p. 264). Tomsk: Radio and Communications.

[11] Klevakin, V. A., Polivanov, A. Y. (2010). Systems of technical vision in industrial robotics. *Mechatronics, Automation, Control*, 9, 26-36.

[12] Gorinov, A. N., Yakovchenko, S. I. (2017). Selection of parametrically specified objects in a low-resolution image. *Reports of TUSUR*, 20(2), 88-90.

[13] International Telecommunication Union. (2015). *Recommendation ITU-R BT.709-6. Parameter values for the HDTV standards for production and international programme exchange*.

[14] Gonzalez, R., Woods, R. (2005). *Digital image processing* (p. 1072). Moscow: Technosphere.

[15] Otsu, N. (2009). A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man and Cybernetics, 9(1), 62-66. https://doi.org/10.1109/TSMC.1979.4310076.