Code 1:  
  
This code implements a method for removing haze from images using a Markov Random Field (MRF)-based dehazing algorithm. Let's break down the main components of the code:

1. Compute Dark Channel Function (‘compute\_dark\_channel’): This function computes the dark channel prior of the input image. The dark channel prior represents the minimum intensity value in a local patch across all color channels. It's used as a prior for estimating the transmission of haze.

2. Estimate Transmission Function (‘estimate\_transmission’): This function estimates the transmission map of the hazy image using the dark channel prior. The transmission map represents the proportion of light that is not scattered or absorbed by the haze in each pixel.

3. Refine Transmission Function (‘refine\_transmission’): This function refines the transmission map using a Markov Random Field (MRF) approach. It iteratively updates the transmission map based on the observed hazy image and the estimated transmission map.

4. Restore Image Function (‘restore\_image’): This function restores the hazy image using the refined transmission map. It calculates the restored image by dividing the observed hazy image by the transmission map and then adding back the atmospheric light.

5. Parameters: The code sets several parameters:

- ‘omega’: A parameter used to balance haze removal and image appearance.

- ‘beta’: A parameter used to control the strength of refinement in the MRF-based dehazing.

- ‘patch\_size’: The size of the patch used for computing the dark channel prior.

- ‘mrf\_iteration’: The number of iterations for refining the transmission map using the MRF approach.

- ‘atmosphere’: The color of atmospheric light, which is subtracted from the image during restoration.

6. Loading Image: The code loads a hazy image from a specified file path using OpenCV's ‘imread’ function.

7. Restoration: The hazy image is then restored using the ‘restore\_image’ function with the specified parameters.

8. Display: The code prints the intensity values of the hazy and restored images and displays them side by side using Matplotlib.

The ultimate goal of this code is to improve the visibility and clarity of images that are affected by haze by removing the haze effect.

Code 2:

This code implements a simple haze removal algorithm based on the dark channel prior. Let's break down the main components of the code:

1. Dark Channel Function (‘dark\_channel’): This function computes the dark channel prior of an image. It iterates through each pixel in the image and extracts a local window centered at that pixel. Then, it calculates the minimum intensity value across all color channels within the window and assigns it to the corresponding pixel in the dark channel image.

2. Transmission Estimation Function (‘transmission\_estimation’): This function estimates the transmission map using the dark channel prior. It computes the transmission value for each pixel as \( 1 - \omega \times \text{dark channel value} \), where \( \omega \) is a parameter controlling the haze removal strength.

3. Haze Removal Function (‘haze\_removal’): This function removes haze from the image. It takes the original image, the estimated transmission map, and the atmosphere color as inputs. For each pixel, it calculates the dehazed pixel value using the formula \( \frac{{\text{image} - \text{atmosphere}}}{{\max(\text{transmission}, 0.1)}} + \text{atmosphere} \). This formula scales the difference between the image and the atmosphere by the inverse of the transmission, ensuring that pixels with low transmission (i.e., heavily affected by haze) receive less correction.

4. Main Function (‘main’): This function is the entry point of the script. It loads the input image, computes the dark channel, estimates the transmission, estimates the atmosphere color, performs haze removal, and displays the original and dehazed images.

5. Loading Image: The code loads a hazy image from a specified file path using OpenCV's ‘imread’ function.

6. Display: The code displays the original and dehazed images side by side using Matplotlib.

Overall, this algorithm provides a simple yet effective way to remove haze from images using the dark channel prior. Adjusting parameters such as the window size and \( \omega \) can fine-tune the results according to specific requirements.

Code 3:  
  
This code implements a haze removal algorithm using the dark channel prior and guided filter. Here's a breakdown of the main components:

1. Dark Channel Function (‘dark\_channel’): This function computes the dark channel prior of an image by iterating over each pixel and calculating the minimum intensity value within a local window centered around that pixel.

2. Transmission Estimation Function (‘transmission\_estimation’): This function estimates the transmission map using the dark channel prior. The transmission value for each pixel is calculated as (1−𝜔)×dark channel value, where 𝜔 is a parameter controlling the strength of haze removal

3. Guided Filter Function (‘guided\_filter’): This function refines the transmission map using a guided filter. It takes the grayscale image and the transmission map as inputs and applies a guided filter to enhance the transmission map.

4. Haze Removal Function (‘haze\_removal’): This function removes haze from the image using the estimated transmission map and atmospheric light. It scales the difference between the image and the atmospheric light by the inverse of the transmission map and adds back the atmospheric light.

5. Remove Haze Function (‘remove\_haze’): This function orchestrates the haze removal process. It first converts the image to grayscale and estimates the atmospheric light by finding the maximum intensity value in the dark channel. Then, it calculates the transmission map, refines it using a guided filter, and removes haze from the image.

6. Main Function (‘main’): This function is the entry point of the script. It loads the input image, removes haze using the ‘remove\_haze’ function, and displays the original and dehazed images.

7. Loading Image: The code loads a hazy image from a specified file path using OpenCV's ‘imread’ function.

8. Display: The code displays the original and dehazed images side by side using Matplotlib.

This algorithm provides a robust approach to haze removal by combining the dark channel prior with guided filtering techniques. Adjusting parameters such as the threshold for transmission map clipping (‘t’), the strength of haze removal (‘omega’), and the radius of the guided filter (‘guided\_radius’) can fine-tune the results for different images and haze conditions.