Photoeffects Documentation

Release 1 alpha

Valentina Kustikova **Pavel Druzhkov**
Kirill Kornyakov **Evgeniy Dolotov** **Artem Screbkov**
Dmitriy Kruchinin **Vadim Levin** **Vlad Vinogradov**

July 07, 2014

Contents

CHAPTER 1

Sepia

Applies sepia effect to image.

int **sepia**(cv::InputArray *src*, cv::OutputArray *dst*)

Parameters

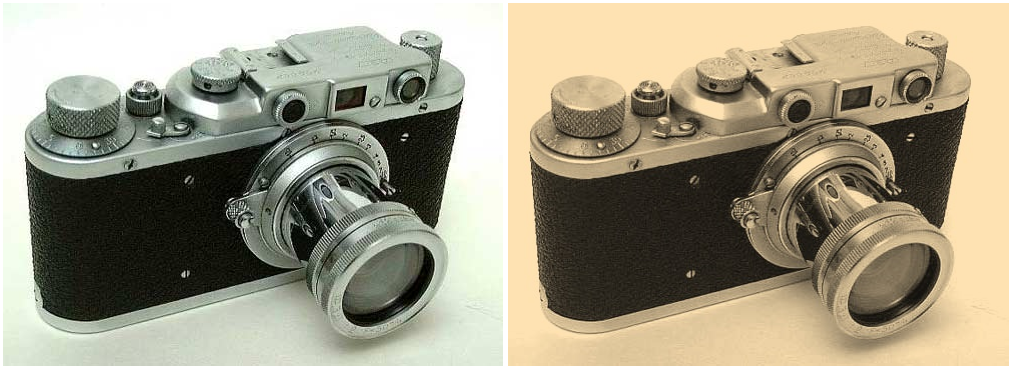
- **src** – Source 8-bit single-channel image.
- **dst** – Destination image of the same size and the same type as **src**.

Returns Error code.

The algorithm.

1. Create 3-channel image, which is interpreted as HSV image.
   (a) 1st channel is the matrix, each element equals **hue** = 19.
   (b) 2nd channel is the matrix, each element equals **saturation** = 78.
   (c) 3rd channel is the matrix of brightness (**src** + **value** = **src** + 20).
2. Convert resulting image to BGR format.

Example.



CHAPTER 2

Antique

Applies antique effect to image.

int **antique**(cv::InputArray *src*, cv::InputArray *texture*, cv::OutputArray *dst*)

Parameters

- **src** – RGB image.
- **texture** – RGB image that is overlaid with the **src** image as a texture of an old photo.
- **dst** – Destination image of the same size and the same type as **src**.

Returns Error code.

The algorithm.

1. Resize texture to match the source image size.
2. Apply sepia matrix transform to the **src**.
3. Calculate the weighted sum of the **texture** and **src**.
4. Convert resulting image to the same color format as **src**.

Example.

CHAPTER 3

Vignette

Makes the edges of the photo less bright, creating an oval frame around its central part.

int **vignette**(cv::InputArray *src*, cv::OutputArray *dst*, cv::Size *rect*)

Parameters

- **src** – Source 8-bit three-channel image.
- **dst** – Destination image of the same size and the same type as **src**.
- **rect** – Size of rectangle describes an ellipse, whose center is at the center image.

Returns Error code.

The algorithm.

1. Create a new 3-channel image, which is interpreted as BGR image.
2. For every channel calculate $dist$, $radius\_ellipse$ and $coefficient$, where $dist$ is a distance between the current pixel and the center of image; $radius\_ellipse$ is a radius of ellipse in the this point which belongs the same line as a pixel; $coefficient$ is a number which multiplies the intensity channel.

   The $coefficient$ is calculated by the following formula:

   $$coefficient = 1 - ((dist - radius\_ellipse)/(max\_radius - radius\_ellipse)),$$

   where max_radius is a distance between the pixel (0, img_src.cols) and the center image.
3. Convert image to a BGR format.

Example.

CHAPTER 4

Matte

Increases the brightness of peripheral pixels.

int **matte**(cv::InputArray *src*, cv::OutputArray *dst*, cv::Point *firstPoint*, cv::Point *secondPoint*, float *sigma*)

Parameters

- **src** – RGB image.
- **dst** – Destination image of the same size and the same type as **src**.
- **firstPoint** – The first point for creating ellipse.
- **secondPoint** – The second point for creatin ellipse.
- **sigma** – The deviation in the Gaussian blur effect.

Returns Error code.

The algorithm.

1. Create new image with white background for mask.
2. Draw black ellipse inscribed in a rectangle that is defined by two opposite corner points (**firstPoint** and **secondPoint**) on the mask image. It's a meaning part.
3. Apply gaussian blur to the meaning part to make fade effect.
4. Convolve mask with the image.
5. Convert resulting image to the same color format as **src**.

Example.

sigma = 25.

CHAPTER 5

Edge Blur

Blurs the edges of an image, keeping center in focus.

int **edgeBlur**(InputArray *src*, OutputArray *dst*, int *indentTop*, int *indentLeft*)

Parameters

- **src** – RGB image.
- **dst** – Destination image of the same size and the same type as **src**.
- **indentTop** – The indent from the top and the bottom of the image. It will be blured.
- **indentLeft** – The indent from the left and right side of the image. It will be blured.

Returns Error code.

The algorithm.

The amount of blurring is defined by the gaussian filter's kernel size and standard deviation and depends on the weighted distance from the center of the image:

$$d(x,y) = \frac{(x-a)^2}{(a-indentLeft)^2} + \frac{(y-b)^2}{(b-indentTop)^2},$$

where $a = \frac{src_{width}}{2}, b = \frac{src_{height}}{2}$. For each pixel (x,y) of the image, if the distance $d(x,y)$ is greater than 1, gaussian filter with center at (x,y), kernel size $d(x,y)$ and standard deviation $(radius - 0.5)$ is applied, otherwise **src** image pixel is left unchanged. Border pixels are replicated to fit the kernel size.

Example.

indentTop = 90, **indentLeft** = 90.

CHAPTER 6

Fade Color

Applies color fade effect to image.

int **`fadeColor`**(InputArray *src*, OutputArray *dst*, Point *startPoint*, Point *endPoint*)

Parameters

- **src** – Grayscale or RGB image.
- **dst** – Destination image of the same size and the same type as **src**.
- **startPoint** – Initial point of direction vector for color fading.
- **endPoint** – Terminal point of direction vector for color fading.

Returns Error code.

The algorithm.

1. Determine the coordinates of the vector by two points **(startPoint, endPoint)** .
2. Determine the line which is perpendicular to vector and is passing through **startPoint**.
3. Find the most distant point from the line.
4. For each pixel located at one side from the line defined by the direction of the vector, change the value of each channel by the following formula:

   newValue = (1-a) * oldValue + a * 255, a = distance / maxDistance.
5. Save this matrix as image in same format.

Example.

Boost Color

Makes colors more saturated.

int **boostColor**(cv::InputArray *src*, cv::OutputArray *dst*, float *intensity=0.0f*)

Parameters

- **src** – RGB image.
- **dst** – Destination image of the same size and the same type as **src**.
- **intensity** – Effect intensity, must be real number from 0.0 to 1.0.

Returns Error code.

The algorithm.

1. Create 3-channel image, which is interpreted as HLS image
2. Each element of the 3rd (Saturation) channel increases by **intensity**
3. Save this matrix as RGB image

Example.

CHAPTER 8

Tint

Add a tint to the image.

int **tint**(cv::InputArray *src*, cv::OutputArray *dst*, const cv::Vec3b& *colorTint*, float *density*)

Parameters

- **src** – Source 8-bit 3-channel (RGB) image.
- **dst** – Destination image of the same size and the same type as **src**.
- **colorTint** – It's a bearing color. All color of the image **src** will be shifted to it.
- **density** – Float value between 0 and 1, defines a range of shift to the colorTint.

Returns Error code.

The algorithm:

Calculate new values by the formula:

$$dst(x, y) = density \cdot src(x, y) + (1 - density) \cdot colorTint$$

Example:

density = 0.1, **colorTint** = Vec3b(255, 255, 0), i.e. cyan.

CHAPTER 9

Warmify

Increases saturation of red and yellow tones, making photographs more warm, sunset view.

int **warmify**(cv::InputArray *src*, cv::OutputArray *dst*, uchar *delta=30*)

Parameters

- **src** – Source 8-bit three-channel image.
- **dst** – Destination image of the same size and the same type as **src**.
- **delta** – Value by which saturation of warm colors is increased.

Returns Error code.

The algorithm.

1. Create 3-channel image, which is interpreted as BGR image.
   (a) 1st channel is the matrix, each element equals **blue** = blue_src.
   (b) 2nd channel is the matrix, each element equals **green** = green_src + delta.
   (c) 3rd channel is the matrix, each element equals **red** = red_src + delta.
2. Save this matrix as BGR image.

Example.

CHAPTER 10

Glow

Applies glow effect to the initial image

int **glow**(cv::InputArray *src*, cv::OutputArray *dst*, int *radius=0*, float *intensity=0.0f*)

Parameters

- **src** – RGB image.
- **dst** – Destination image of the same size and the same type as **src**.
- **radius** – Radius of box filter kernel, must be positive integer number
- **intensity** – Effect intensity, must be real number from 0.0 to 1.0

Returns Error code.

The algorithm.

1. Create the copy of the source image
2. Apply box filter for this copy with deviation equal **sigma**
3. Create new 3-channel image, each channel of the matrix calculates by the following formula:

$$C = \begin{cases} 2 * A * B, & \text{if } B <= 0.5 \\ 1 - 2 * (1 - A) * (1 - B), & \text{otherwise} \end{cases}$$

   where A is the pixel's tonal value that lies in the blurred image, B is the pixel's tonal value that lies in the source image, C is the tonal value of the blended pixel.

4. Create new 3-channel image, each channel of the matrix calculates by the following formula:

$$C = intensity * A + (1 - intensity) * B$$

   where A is the pixel's tonal value that lies in the previous blended image, B is the pixel's tonal value that lies in the source image, C is the tonal value of the blended pixel.

5. Save previous image as RGB image

Example.

Film Grain

Applies film grain effect to the initial image.

int **`filmGrain`**(InputArray *src*, OutputArray *dst*, int *grainValue*, RNG& *rng*)

Parameters

- **src** – Grayscale or RGB image.
- **dst** – Destination image of the same size and the same type as **src**.
- **grainValue** – Degree of graininess. 8 is default value.
- **rng** – Random number generator. cv::theRNG() is default value.

Returns Error code.

The algorithm.

1. Create matrix with noise.
2. Add noise to image.

Example.