cv::Mat image = cv::imread("image.jpg"); for (int y = 0; y < image.rows; y++) { for (int x = 0; x < image.cols; x++) { cv::Vec3b &pixel = image.at<cv::Vec3b>(y, x); // BGR formátum pixel[0] = 255; // Példa: módosítjuk a kék csatornát } }

cv::MatIterator\_<cv::Vec3b> it, end; for (it = image.begin<cv::Vec3b>(), end = image.end<cv::Vec3b>(); it != end; ++it) { (\*it)[0] = 255; // Példa: módosítjuk a kék csatornát }

cv::Mat gray\_image; cv::cvtColor(image, gray\_image, cv::COLOR\_BGR2GRAY);

cv::Mat blurred\_image; cv::GaussianBlur(image, blurred\_image, cv::Size(5, 5), 0);

cv::Mat gray\_image, equalized\_image; cv::cvtColor(image, gray\_image, cv::COLOR\_BGR2GRAY); cv::equalizeHist(gray\_image, equalized\_image);

cv::Mat gray\_image, binary\_image;

cv::cvtColor(image, gray\_image, cv::COLOR\_BGR2GRAY);

cv::threshold(gray\_image, binary\_image, 128, 255, cv::THRESH\_BINARY);

cv::Mat eroded\_image;

cv::erode(binary\_image, eroded\_image, cv::Mat(), cv::Point(-1, -1), 1);

cv::Mat dilated\_image;

cv::dilate(binary\_image, dilated\_image, cv::Mat(), cv::Point(-1, -1), 1);

cv::Mat grad\_x, grad\_y;

cv::Sobel(gray\_image, grad\_x, CV\_16S, 1, 0, 3);

cv::Sobel(gray\_image, grad\_y, CV\_16S, 0, 1, 3);

cv::convertScaleAbs(grad\_x, grad\_x);

cv::convertScaleAbs(grad\_y, grad\_y);

cv::Mat laplacian\_image;

cv::Laplacian(gray\_image, laplacian\_image, CV\_16S, 3);

cv::convertScaleAbs(laplacian\_image, laplacian\_image);

cv::Mat edges;

cv::Canny(gray\_image, edges, 100, 200);