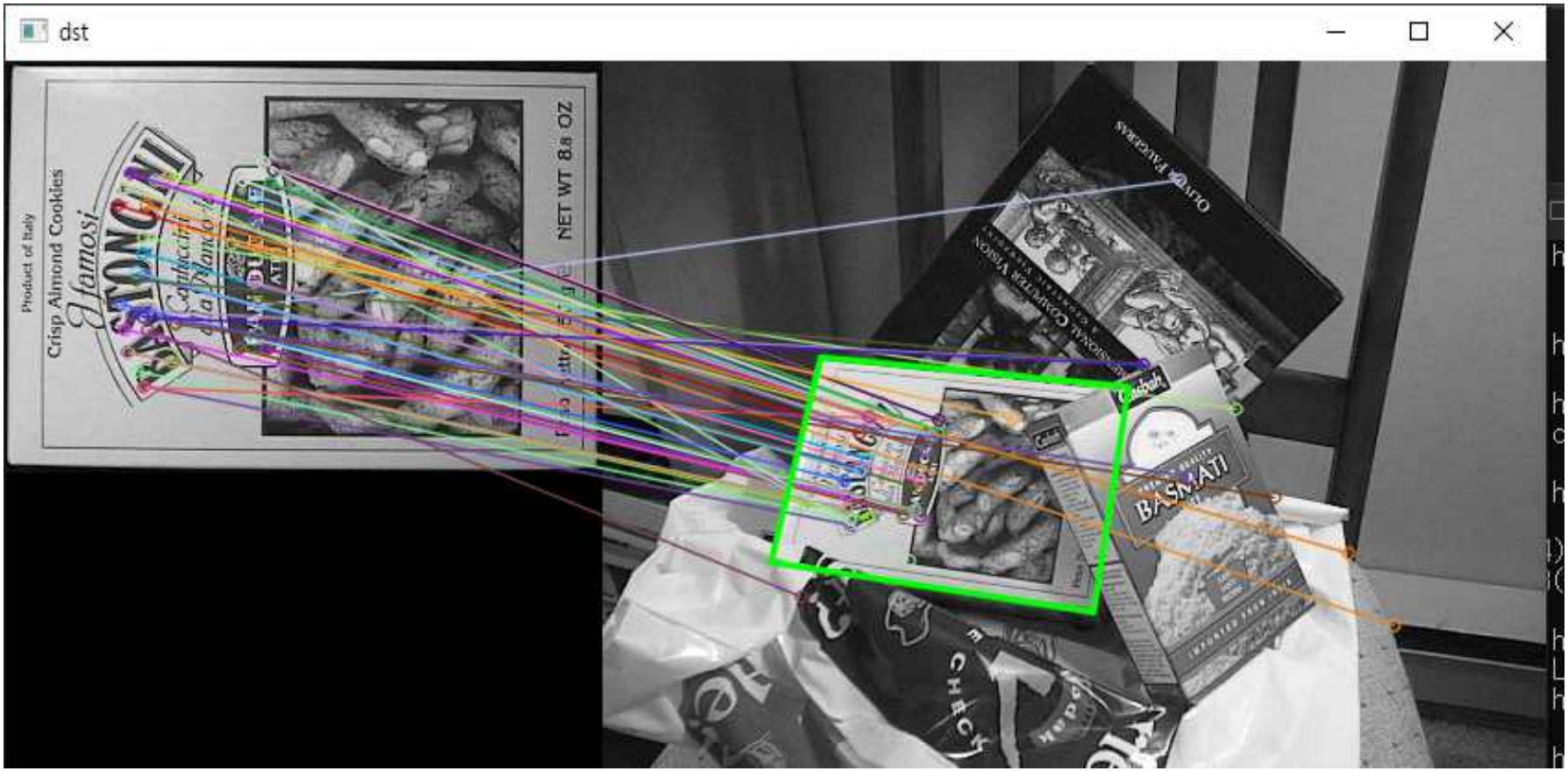
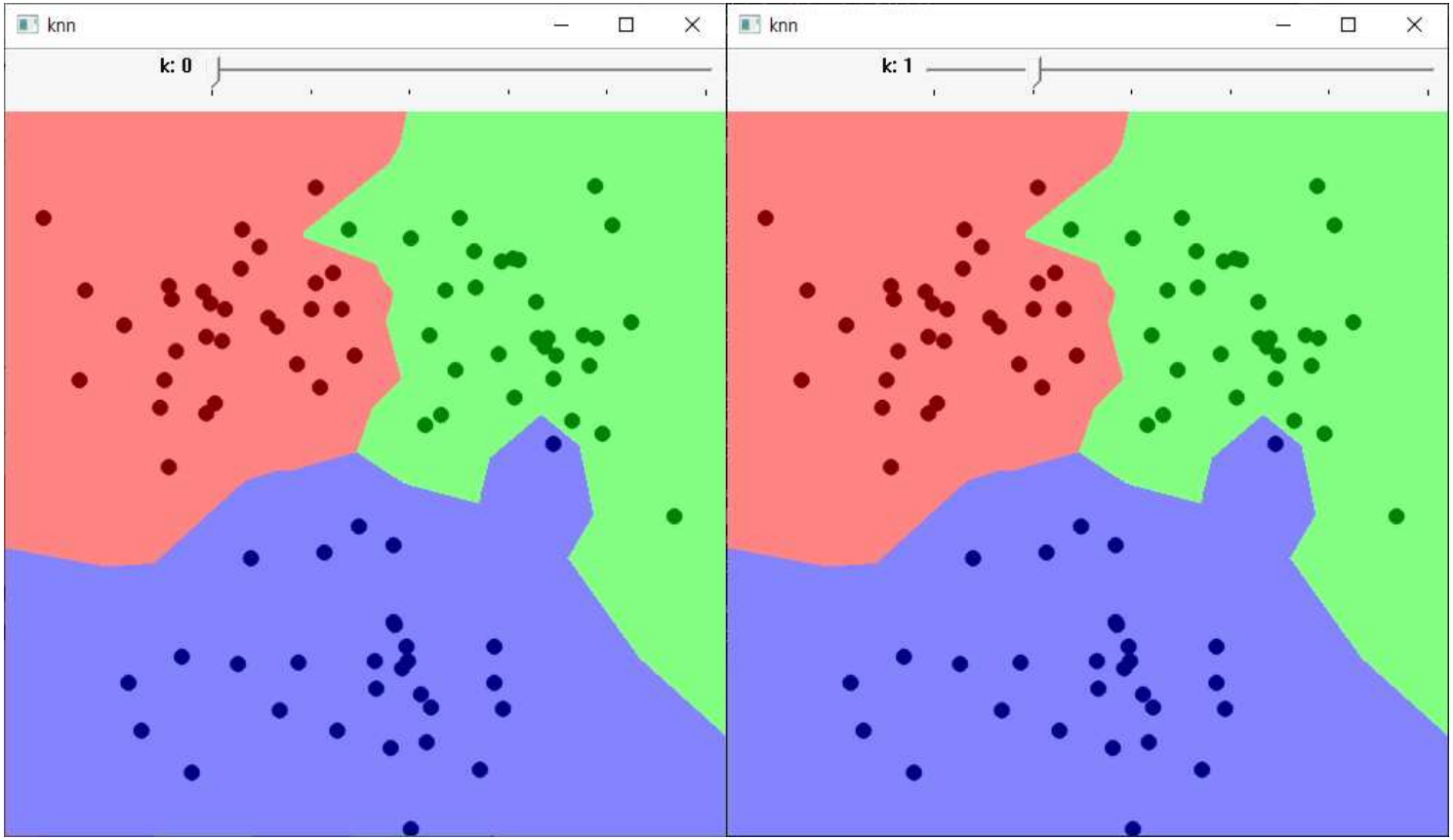
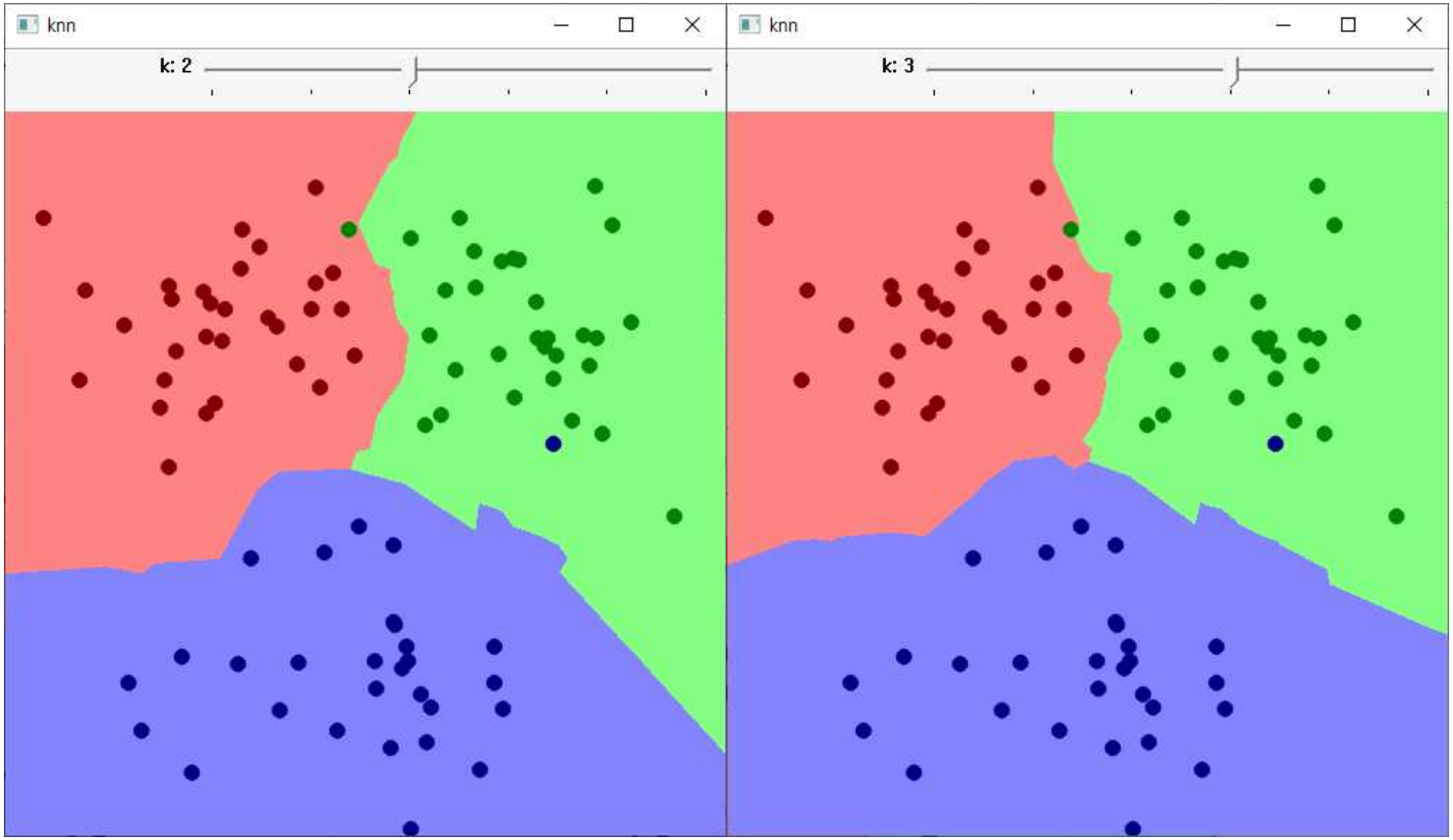


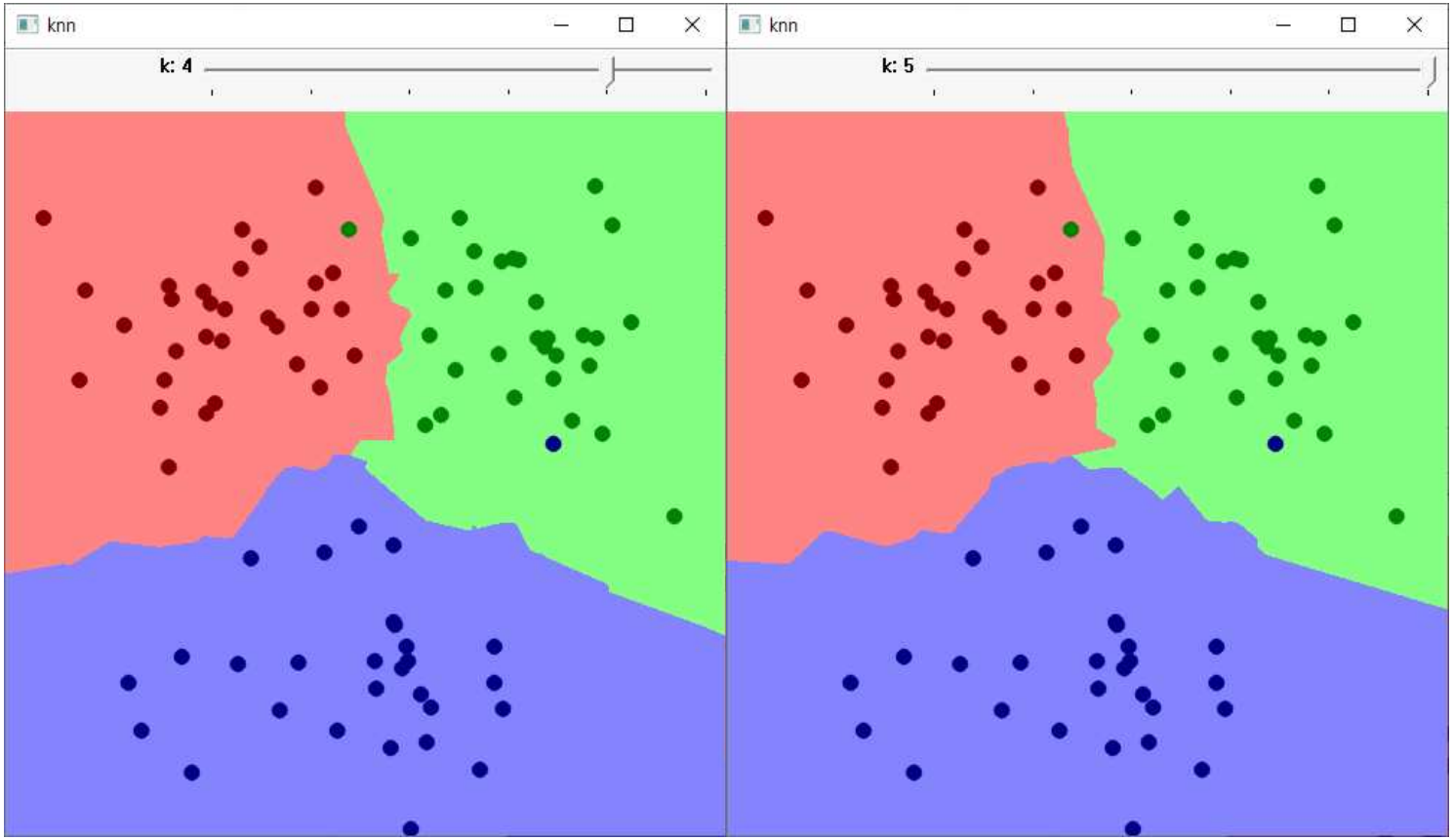
1) 코드 14-8 Homography and Matching 예제 실습

```
void find_homography()
{
	Mat src1 = imread("box.png", IMREAD_GRAYSCALE);
	Mat src2 = imread("box_in_scene.png", IMREAD_GRAYSCALE);
	if (src1.empty() || src2.empty()) {
		cerr << "Image load failed!" << endl;
		return; }
	Ptr<Feature2D> orb = ORB::create();
	vector<KeyPoint> keypoints1, keypoints2;
	Mat desc1, desc2;
	orb->detectAndCompute(src1, Mat(), keypoints1, desc1);
	orb->detectAndCompute(src2, Mat(), keypoints2, desc2);
	Ptr<DescriptorMatcher> matcher = BFMatcher::create(NORM_HAMMING);
	vector<DMatch> matches;
	matcher->match(desc1, desc2, matches);
	std::sort(matches.begin(), matches.end());
	vector<DMatch> good_matches(matches.begin(), matches.begin() + 50);

	Mat dst;
	drawMatches(src1, keypoints1, src2, keypoints2, good_matches, dst,
		Scalar::all(-1), Scalar::all(-1), vector<char>(),
		DrawMatchesFlags::NOT_DRAW_SINGLE_POINTS);

	vector<Point2f> pts1, pts2;
	for (size_t i = 0; i < good_matches.size(); i++) {
```

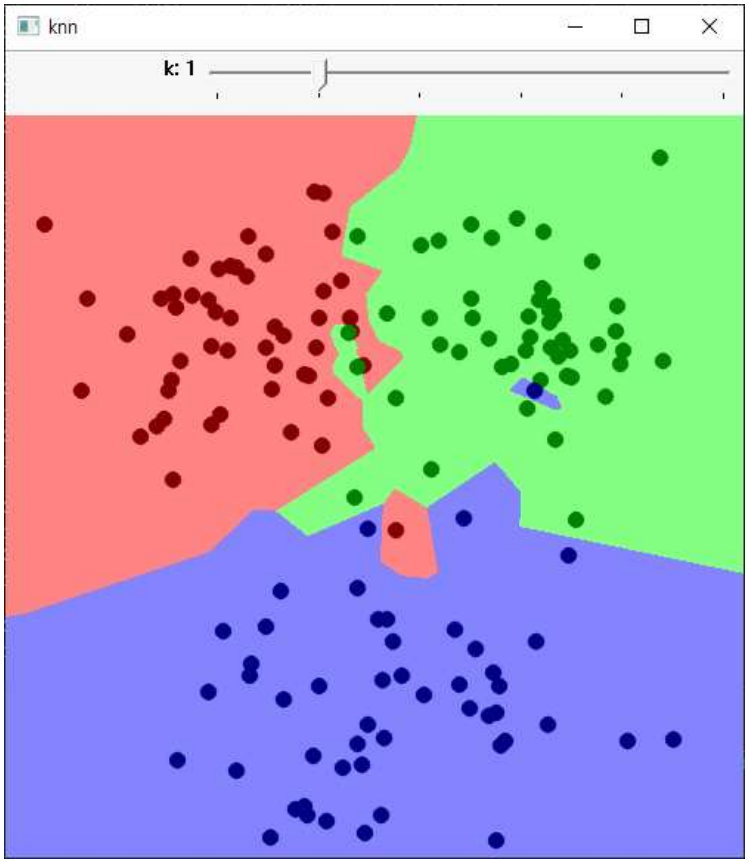
```
            pts1.push_back(keypoints1[good_matches[i].queryIdx].pt);
            pts2.push_back(keypoints2[good_matches[i].trainIdx].pt);
        }

        Mat H = findHomography(pts1, pts2, RANSAC);

        vector<Point2f> corners1, corners2;
        corners1.push_back(Point2f(0, 0));
        corners1.push_back(Point2f(src1.cols - 1.f, 0));
        corners1.push_back(Point2f(src1.cols - 1.f, src1.rows - 1.f));
        corners1.push_back(Point2f(0, src1.rows - 1.f));
        perspectiveTransform(corners1, corners2, H);

        vector<Point> corners_dst;
        for (Point2f pt : corners2) {
            corners_dst.push_back(Point(cvRound(pt.x + src1.cols), cvRound(pt.y)));
        }

        polylines(dst, corners_dst, true, Scalar(0, 255, 0), 2, LINE_AA);

        imshow("dst", dst);

        waitKey();
        destroyAllWindows();

}
```


knn
k: 0
knn
k: 1

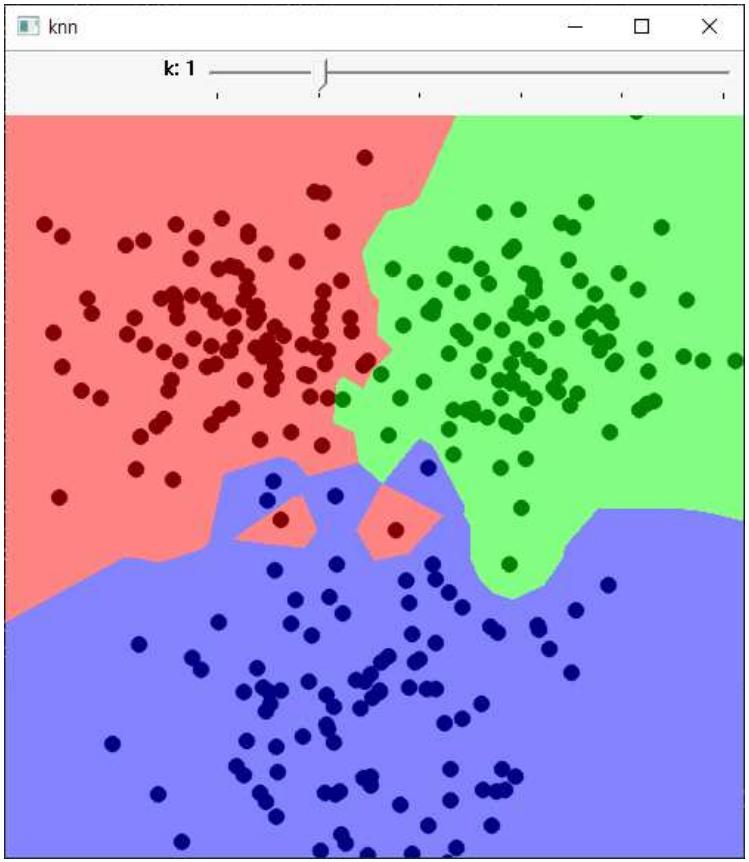

knn
k: 2
knn
k: 3



2) 코드 15-1 K-NN 2D Classification 예제 실습
k값의 변화에 따라 분류가 달라지는 모습을 볼 수 있다.



점 개수를 50개로 늘려본 결과



점 개수를 100개로 늘려본 결과

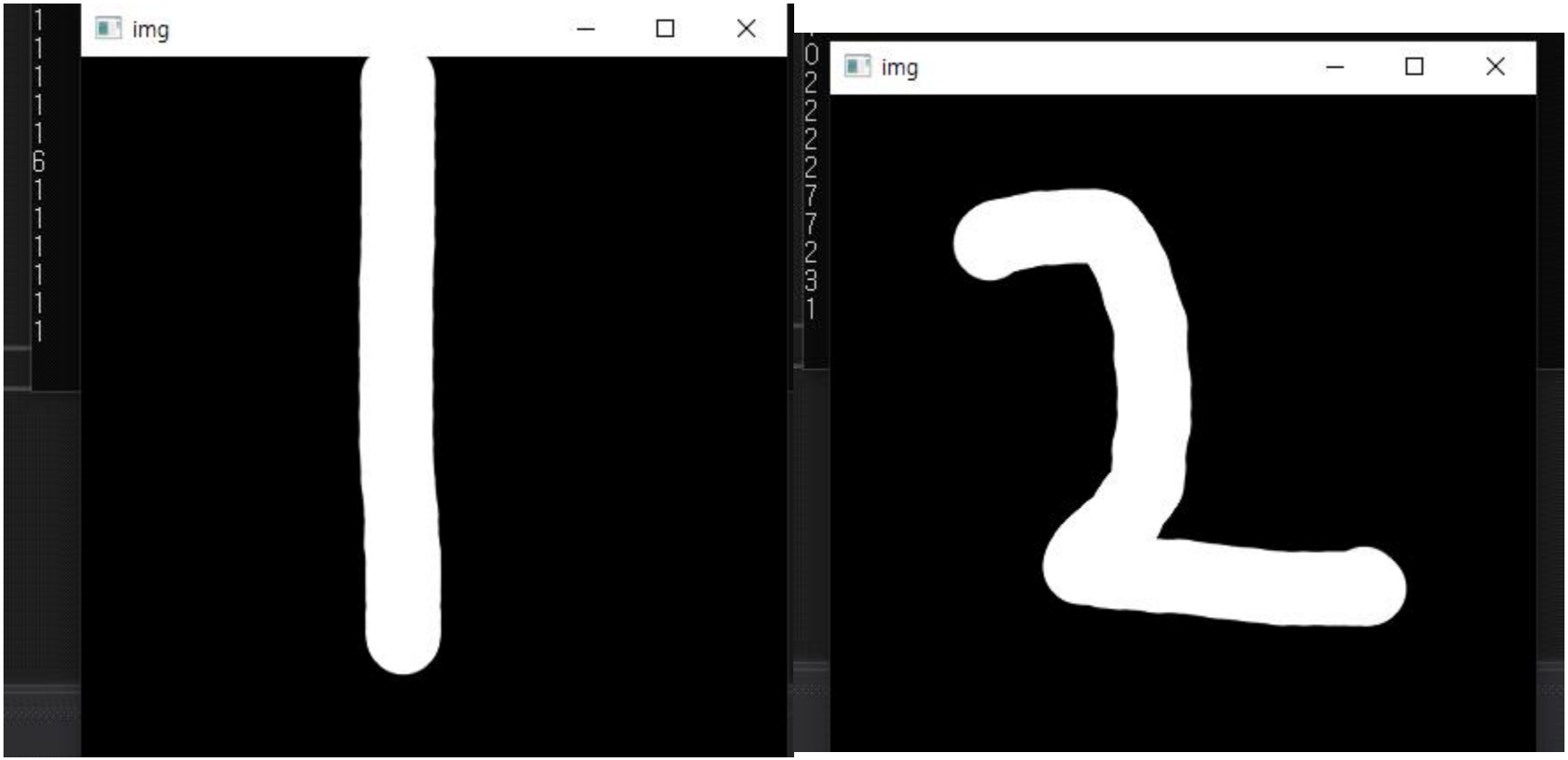
```
Mat img;
Mat train, label;
Ptr<KNearest> knn;
int k_value = 1;

void on_k_changed(int, void*);
void addPoint(const Point&, int cls);
void trainAndDisplay();

int main(){
	img = Mat::zeros(Size(500, 500), CV_8UC3);
	knn = KNearest::create();

	const int NUM = 100;
	Mat rn(NUM, 2, CV_32SC1);

	randn(rn, 0, 50);
	for (int i = 0; i < NUM; i++) {
		addPoint(Point(rn.at<int>(i, 0) + 150, rn.at<int>(i, 1) + 150), 0);
	}
	randn(rn, 0, 50);
	for (int i = 0; i < NUM; i++) {
		addPoint(Point(rn.at<int>(i, 0) + 350, rn.at<int>(i, 1) + 150), 1);
	}
	randn(rn, 0, 70);
```

```
	for (int i = 0; i < NUM; i++) {
		addPoint(Point(rn.at<int>(i, 0) + 250, rn.at<int>(i, 1) + 400), 2);
	}


	namedWindow("knn");
	createTrackbar("k", "knn", &k_value, 5, on_k_changed);



	trainAndDisplay();

	waitKey();
	return 0;
}
void on_k_changed(int, void*) {
	if (k_value < 1) k_value = 1;
	trainAndDisplay();
}
void addPoint(const Point& pt, int cls) {
	Mat new_sample = (Mat_<float>(1, 2) << pt.x, pt.y);
	train.push_back(new_sample);

	Mat new_label = (Mat_<int>(1,1) << cls);
	label.push_back(new_label);
}
```

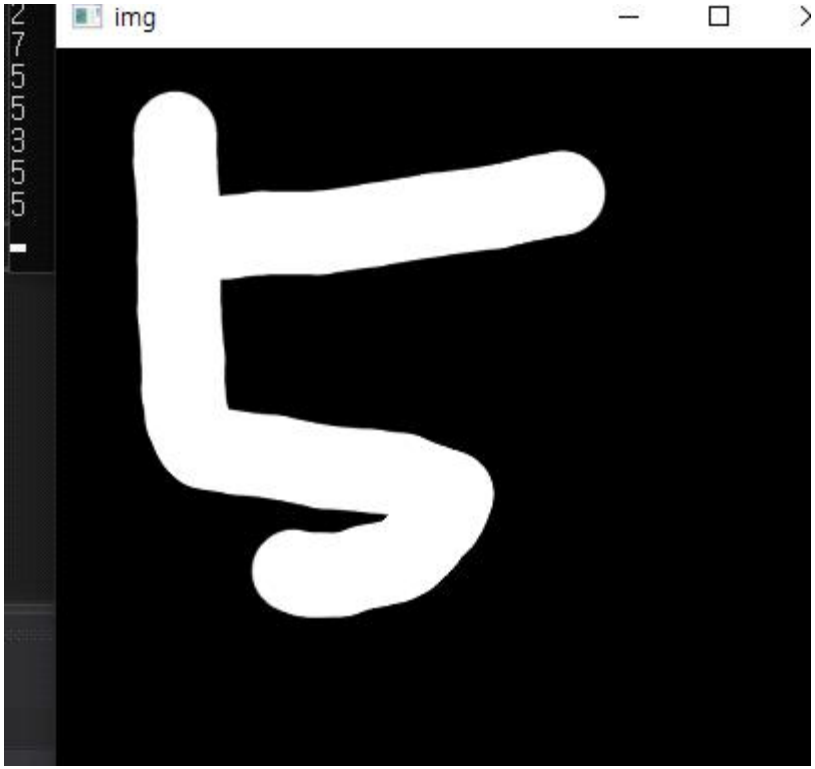
```
void trainAndDisplay() {
	knn->train(train, ROW_SAMPLE, label);
	for (int i = 0; i < img.rows; ++i) {
		for (int j = 0; j < img.cols; ++j) {
			Mat sample = (Mat_<float>(1, 2) << j, i);
			Mat res;
			knn->findNearest(sample, k_value, res);

			int response = cvRound(res.at<float>(0, 0));
			if (response == 0) {
				img.at<Vec3b>(i, j) = Vec3b(128, 128, 255); // R
			}
			else if (response == 1) {
				img.at<Vec3b>(i, j) = Vec3b(128, 255, 128); // G
			}
			else if (response == 2) {
				img.at<Vec3b>(i, j) = Vec3b(255, 128, 128); // B
			}
		}
	}

	for (int i = 0; i < train.rows; i++) {
		int x = cvRound(train.at<float>(i, 0));
		int y = cvRound(train.at<float>(i, 1));
		int l = label.at<int>(i, 0);
```

```
                if (l == 0) {
                        circle(img, Point(x, y), 5, Scalar(0, 0, 128), -1, LINE_AA);
                }
                else if (l == 1) {
                        circle(img, Point(x, y), 5, Scalar(0, 128, 0), -1, LINE_AA);
                }
                else if (l == 2) {
                        circle(img, Point(x, y), 5, Scalar(128, 0, 0), -1, LINE_AA);
                }
        }

        imshow("knn", img);
}
```

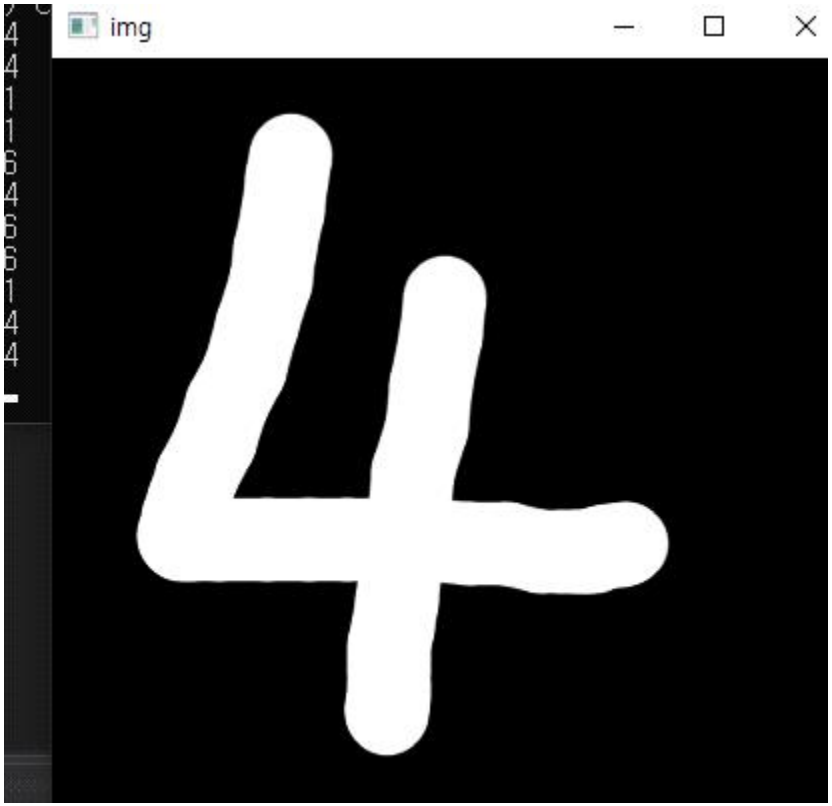
3) 코드 15-2,3,4 K-NN Classification 예제 실습




img
img


img

k값을 5로 바꾼 결과



k값을 1로 바꾼 결과
최초의 3이나 5로 바꿨을 때 보다 4를 더 잘 인식했다.

```
Ptr<KNearest> train_knn() {
	Mat digits = imread("digits.png", IMREAD_GRAYSCALE);

	if (digits.empty()) {
		cerr << "Image load failed!" << endl;
		return 0;
	}

	Mat train_images, train_labels;

	for (int j = 0; j < 50; j++) {
		for (int i = 0; i < 100; i++) {
			Mat roi, roi_float, roi_flatten;
			roi = digits(Rect(i * 20, j * 20, 20, 20));
			roi.convertTo(roi_float, CV_32F);
			roi_flatten = roi_float.reshape(1, 1);

			train_images.push_back(roi_flatten);
			train_labels.push_back(j / 5);
		}
		cout << j / 5 << endl;
	}

	Ptr<KNearest> knn = KNearest::create();
	knn->train(train_images, ROW_SAMPLE, train_labels);
	return knn;
```

```
}
Point ptPrev(-1, -1);

void on_mouse(int event, int x, int y, int flags, void* userdata) {
	Mat img = *(Mat*)userdata;

	if (event == EVENT_LBUTTONDOWN) {
		ptPrev = Point(x, y);
	}
	else if (event == EVENT_LBUTTONUP) {
		ptPrev = Point(-1, -1);
	}
	else if (event == EVENT_MOUSEMOVE && (flags & EVENT_FLAG_LBUTTON)) {
		line(img, ptPrev, Point(x, y), Scalar::all(255), 40, LINE_AA, 0);
		ptPrev = Point(x, y);

		imshow("img", img);
	}
}

int main(){
	Ptr<KNearest> knn = train_knn();

	if (knn.empty()) {
		cerr << "Training failed!" << endl;
		return 0;
```

```
}

Mat img = Mat::zeros(400, 400, CV_8U);

imshow("img", img);
setMouseCallback("img", on_mouse, (void*)&img);

while (true) {
	int c = waitKey(0);

	if (c == 27) {
		break;
	}
	else if (c == ' ') {
		Mat img_resize, img_float, img_flatten, res;

		resize(img, img_resize, Size(20, 20), 0, 0, INTER_AREA);
		img_resize.convertTo(img_float, CV_32F);
		img_flatten = img_float.reshape(1, 1);

		knn->findNearest(img_flatten, 3, res);
		cout << cvRound(res.at<float>(0, 0)) << endl;

		img.setTo(0);
		imshow("img", img);
	}
```

```
        }
    }
```