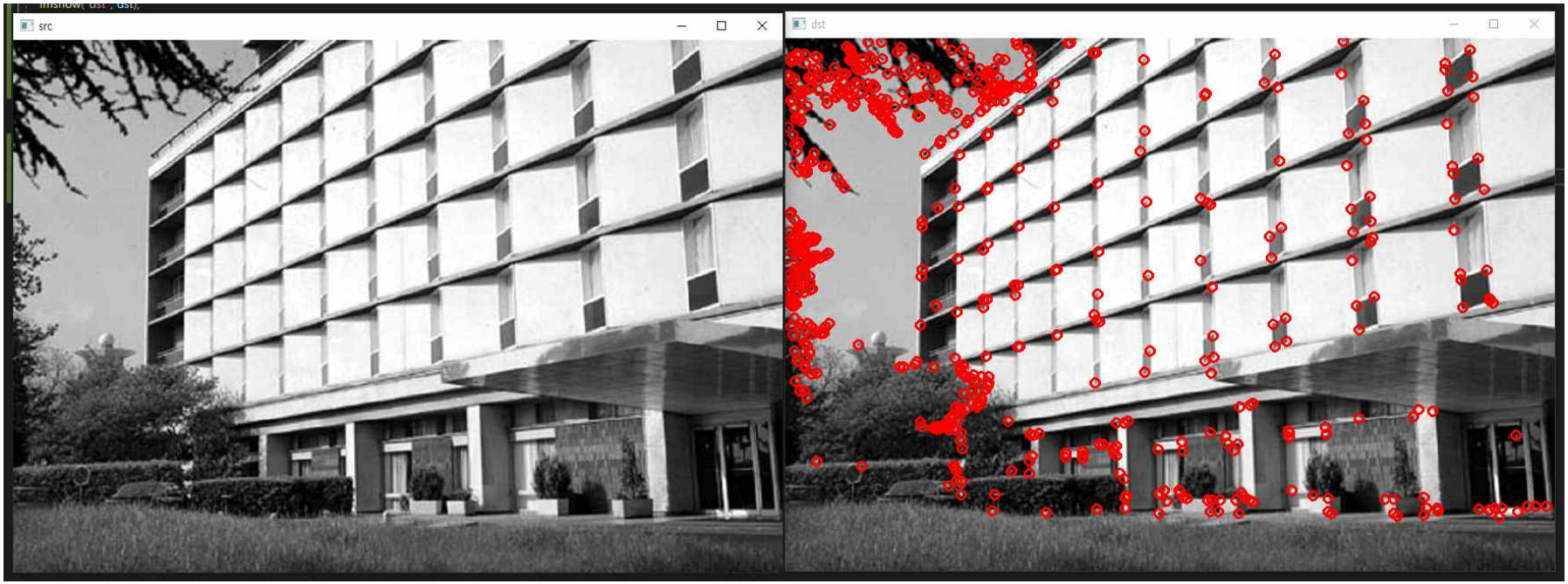
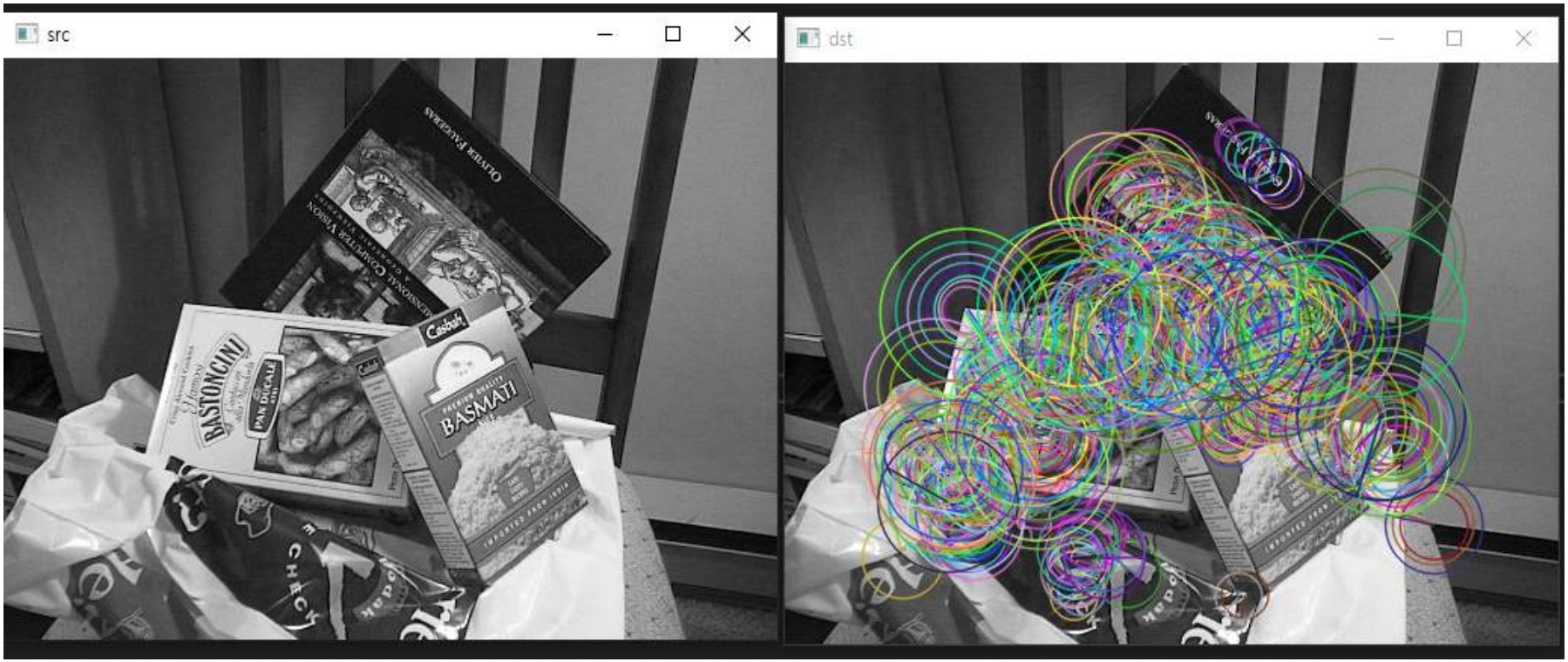


실습14-2

```
void corner_fast() {

	Mat src = imread("building.jpg", IMREAD_GRAYSCALE);
	if (src.empty()) {
		cerr << "Image load failed!" << endl;
		return;
	}
	vector<KeyPoint> keypoints;
```

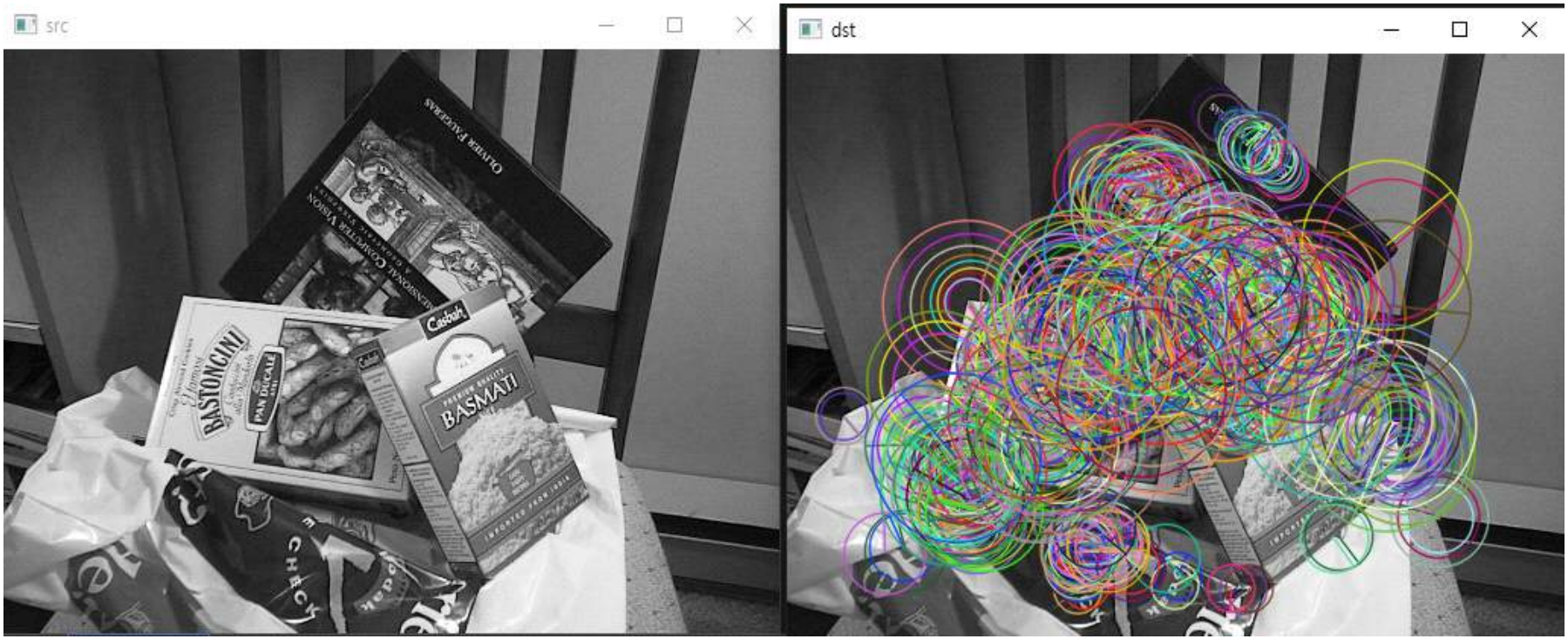
```
        FAST(src, keypoints, 60, true);
        Mat dst; cvtColor(src, dst, COLOR_GRAY2BGR);
        for (KeyPoint kp : keypoints) {
                Point pt(cvRound(kp.pt.x), cvRound(kp.pt.y));
                circle(dst, pt, 5, Scalar(0, 0, 255), 2);
        }
        imshow("src", src);
        imshow("dst", dst);
        waitKey()           ;
        destroyAllWindows();

}
```



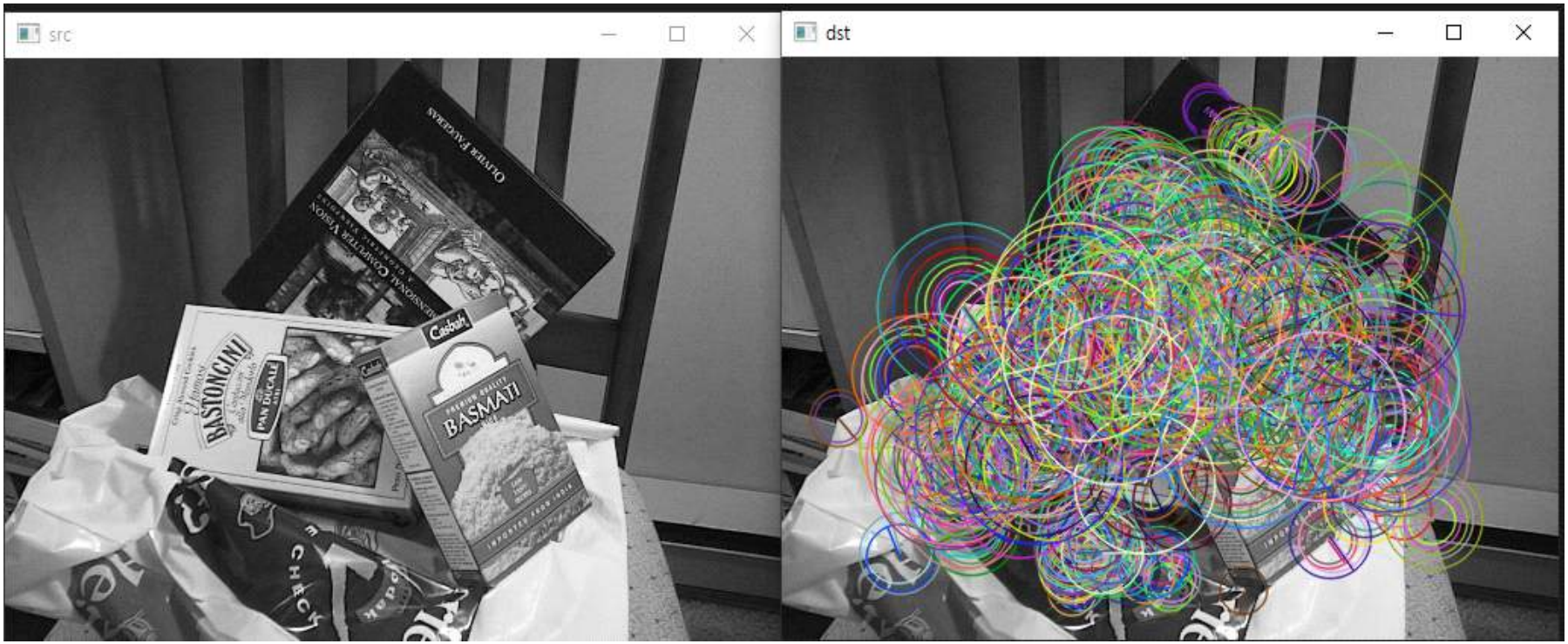
실습 14-4

```
void detect_keypoint() {
	//orb 생성시 파라메터 변경
	Mat src = imread("box_in_scene.png", IMREAD_GRAYSCALE);
	if (src.empty()) {
		cerr << "Image load failed!" << endl;
		return;
	}
```

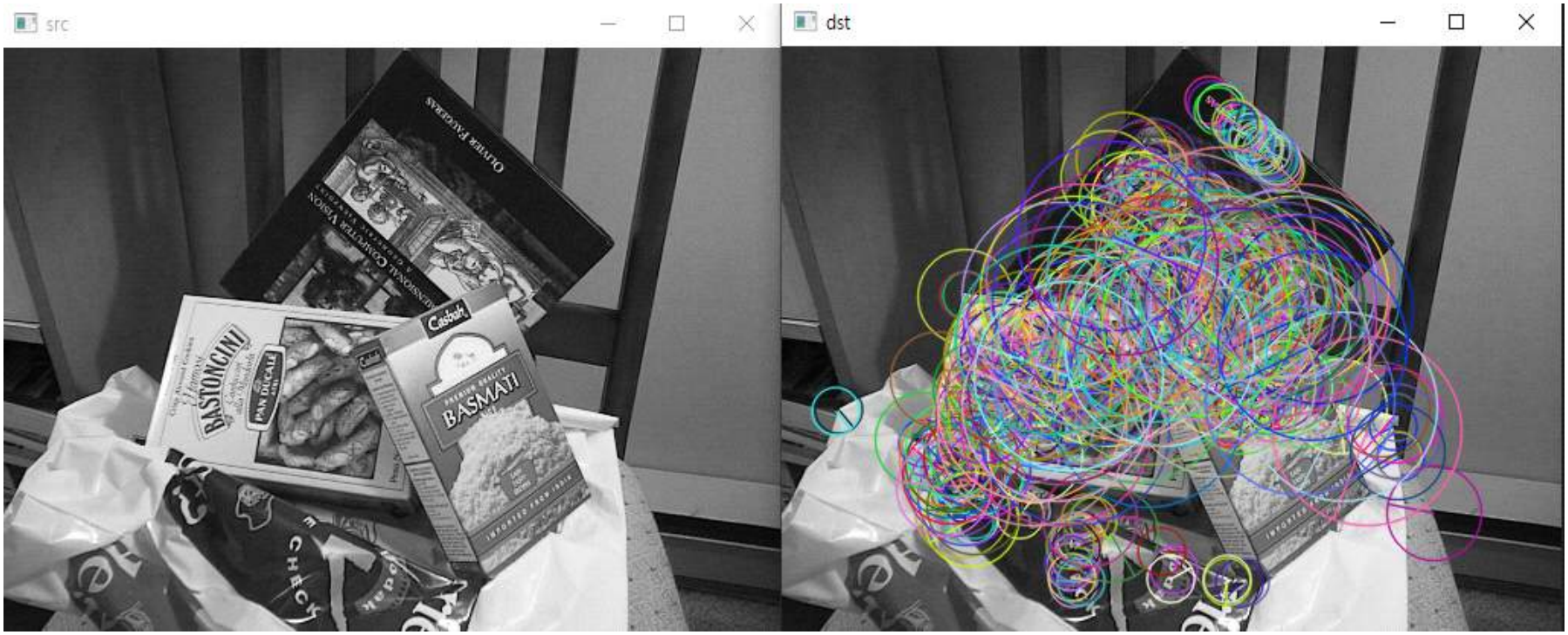
```
    Ptr<Feature2D> feature = ORB::create();

    vector<KeyPoint> keypoints;
    feature->detect(src, keypoints);

    Mat desc;
    feature->compute(src, keypoints, desc);
    cout << "keypoints.size(): " << keypoints.size() << endl;
    cout << "desc.size(): " << desc.size() << endl;
    Mat dst;
    drawKeypoints(src, keypoints, dst, Scalar::all(-1), DrawMatchesFlags::DRAW_RICH_KEYPOINTS);
    imshow("src", src);
    imshow("dst", dst);
    waitKey();
    destroyAllWindows();

}
```



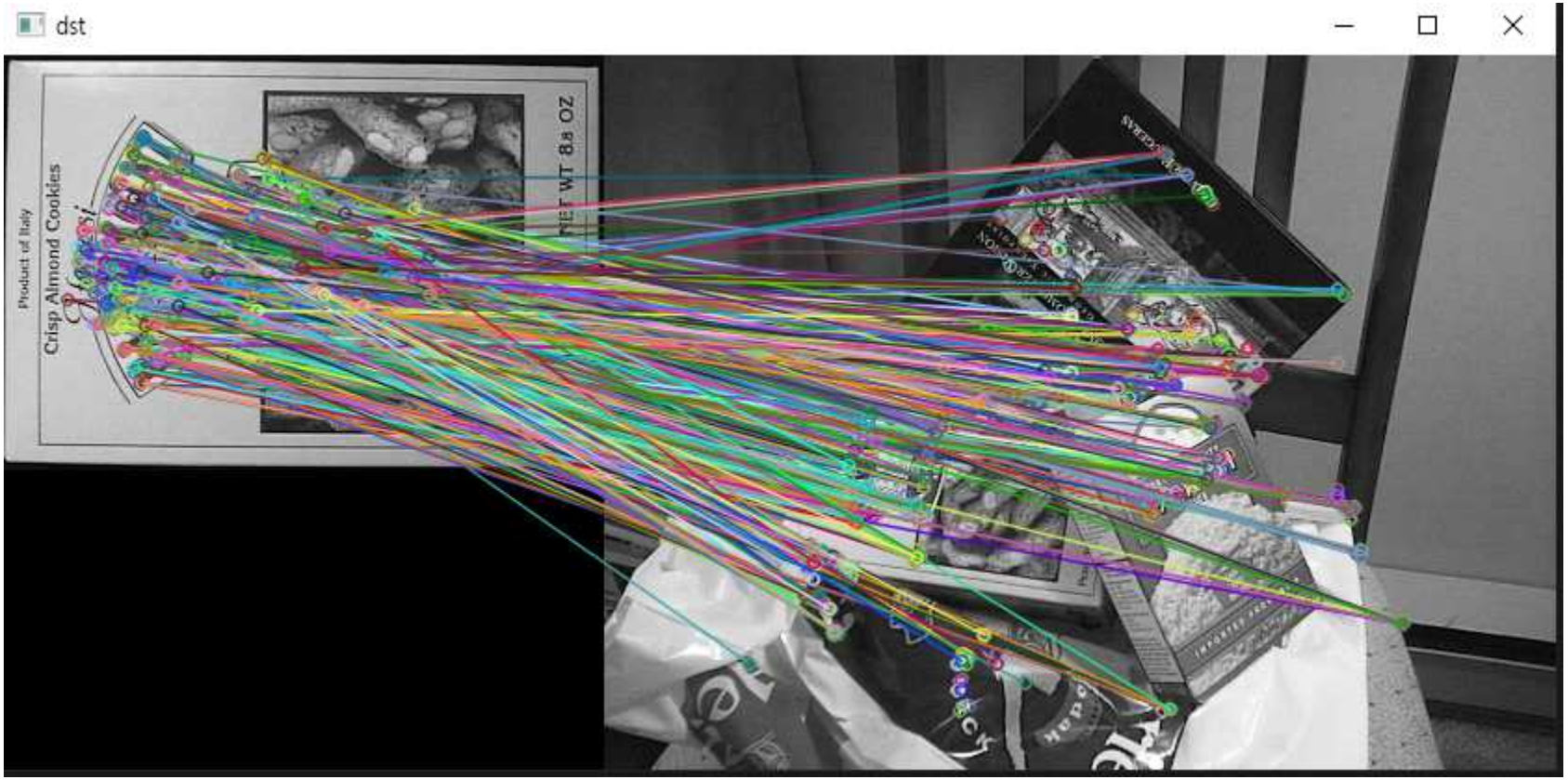
nfeatures를 800으로 변경했을 때 특징점을 더많이 찾음을 알 수 있다



특징점 점수 결정방법을 FAST_SCORE로 변경했을 때 HARRIS_SCORE와 다른 변경점을 찾음을 알 수 있다.



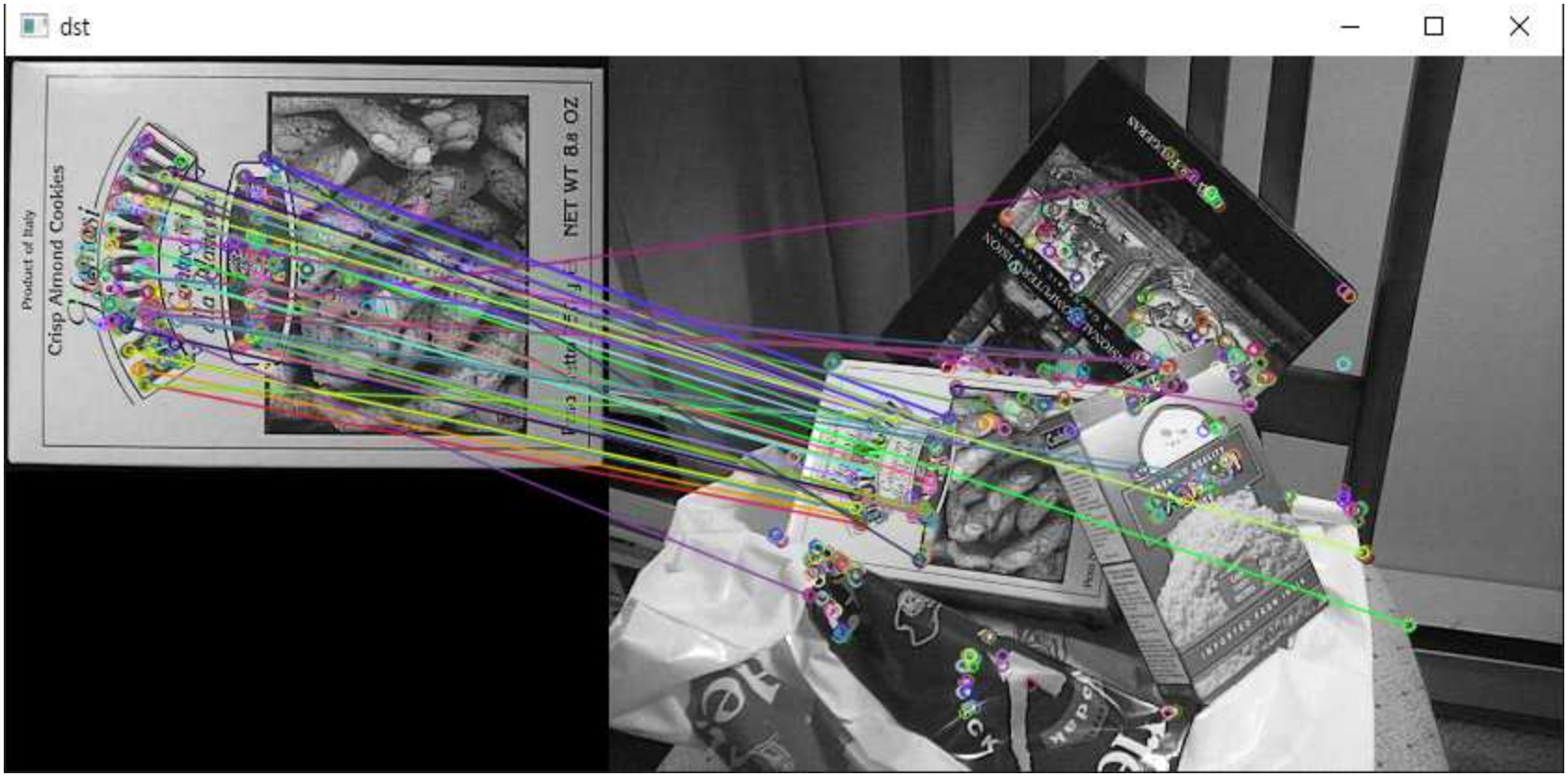
피라미드 생성비율을 2.0f로 바꿨을 때



실습 14-6

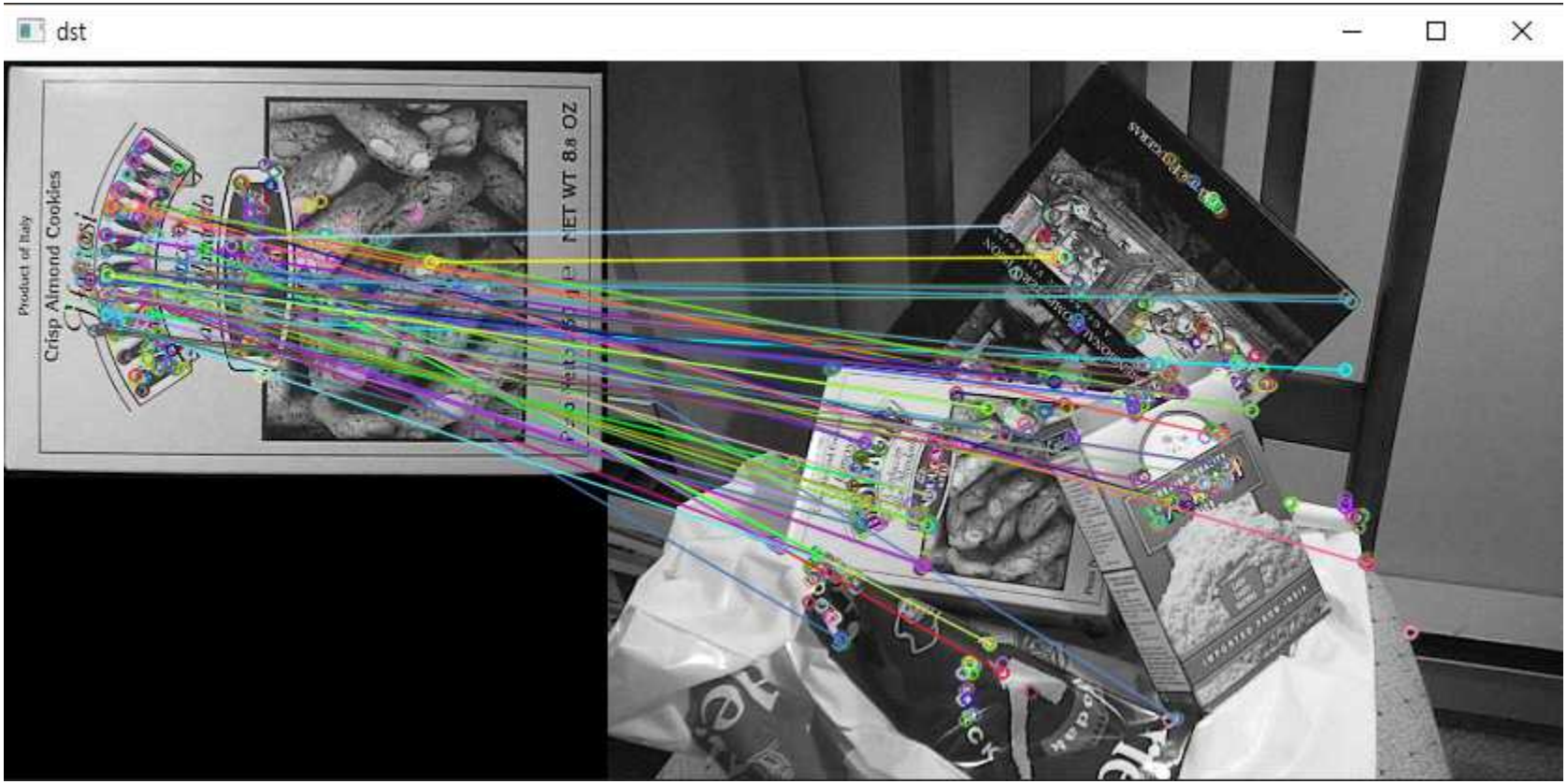
```
void keypoint_matching() {
        Mat src1 = imread("box.png", IMREAD_GRAYSCALE);
        Mat src2 = imread("box_in_scene.png", IMREAD_GRAYSCALE);
```

```
	if (src1.empty() || src2.empty()) {
		cerr << "Image load failed!" << endl;
		return;
	}
	Ptr<Feature2D> feature = ORB::create();
	vector<KeyPoint> keypoints1, keypoints2;
	Mat desc1, desc2;
	feature->detectAndCompute(src1, Mat(), keypoints1, desc1);
	feature->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);
	imshow("dst", dst);
	waitKey();
	destroyAllWindows();

}
```

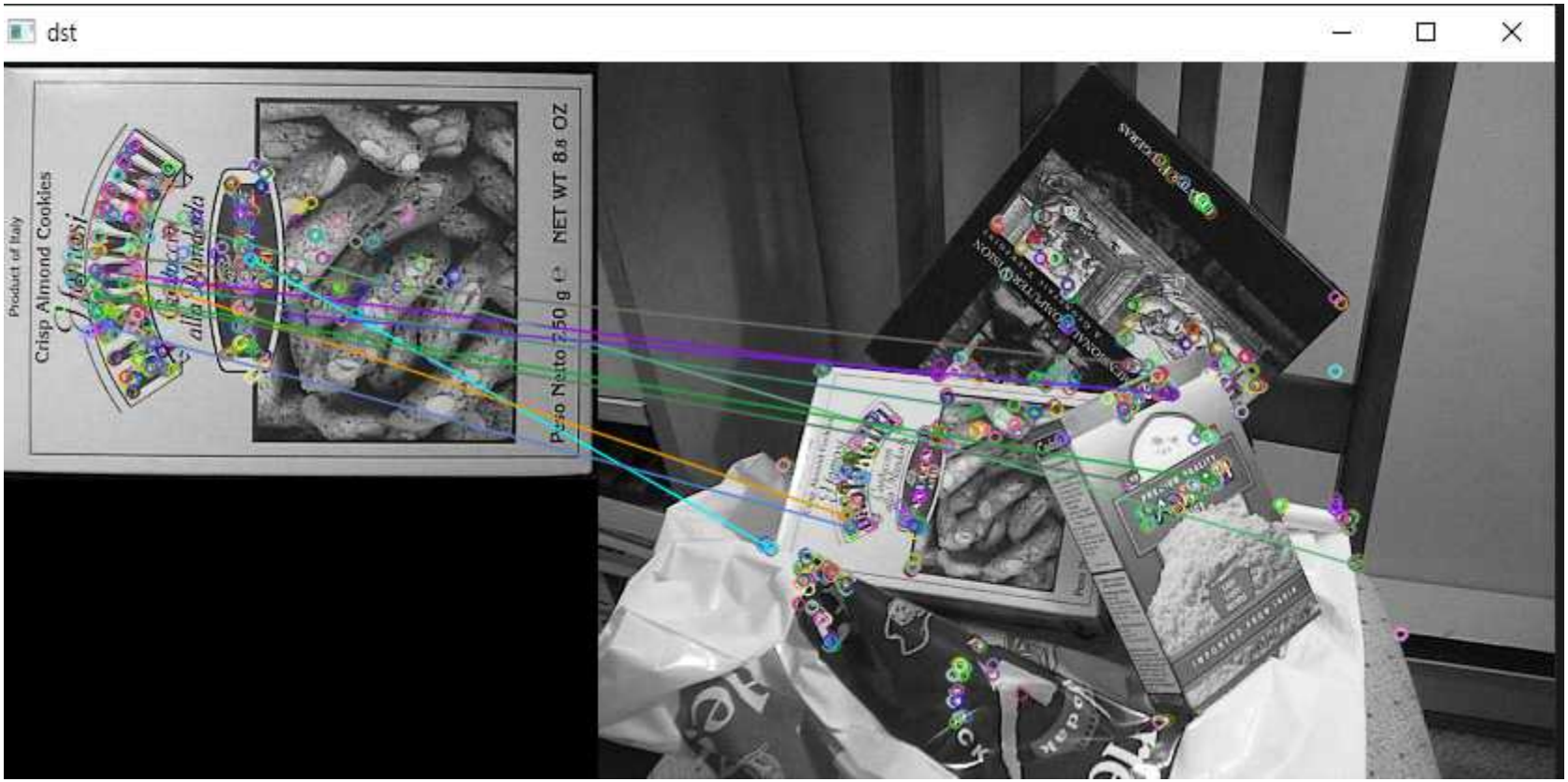


정렬하고 상위 50개의 matching만 추출한 결과물

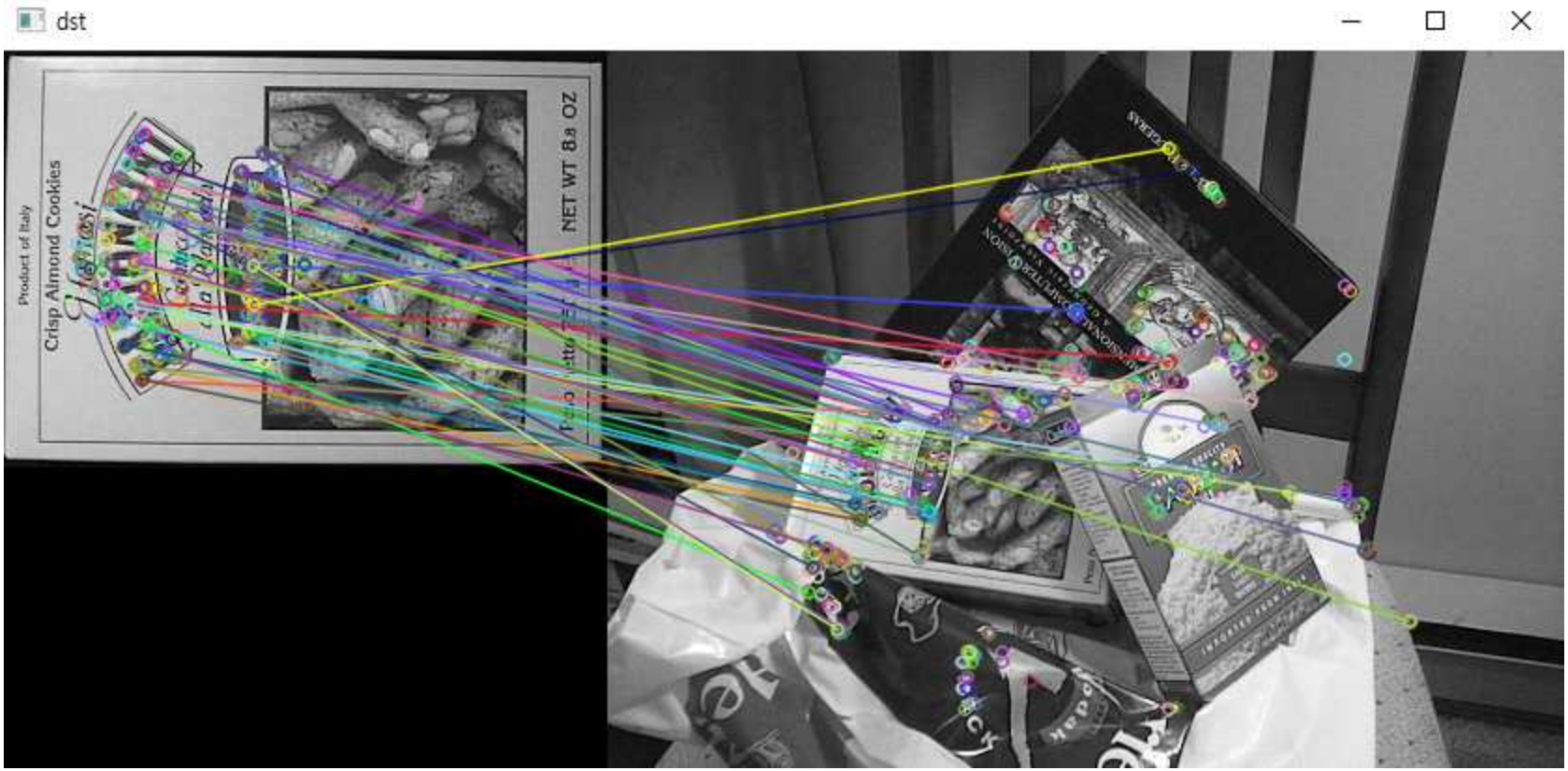
정렬 후 상위 10개의 matching만 추출한 결과물



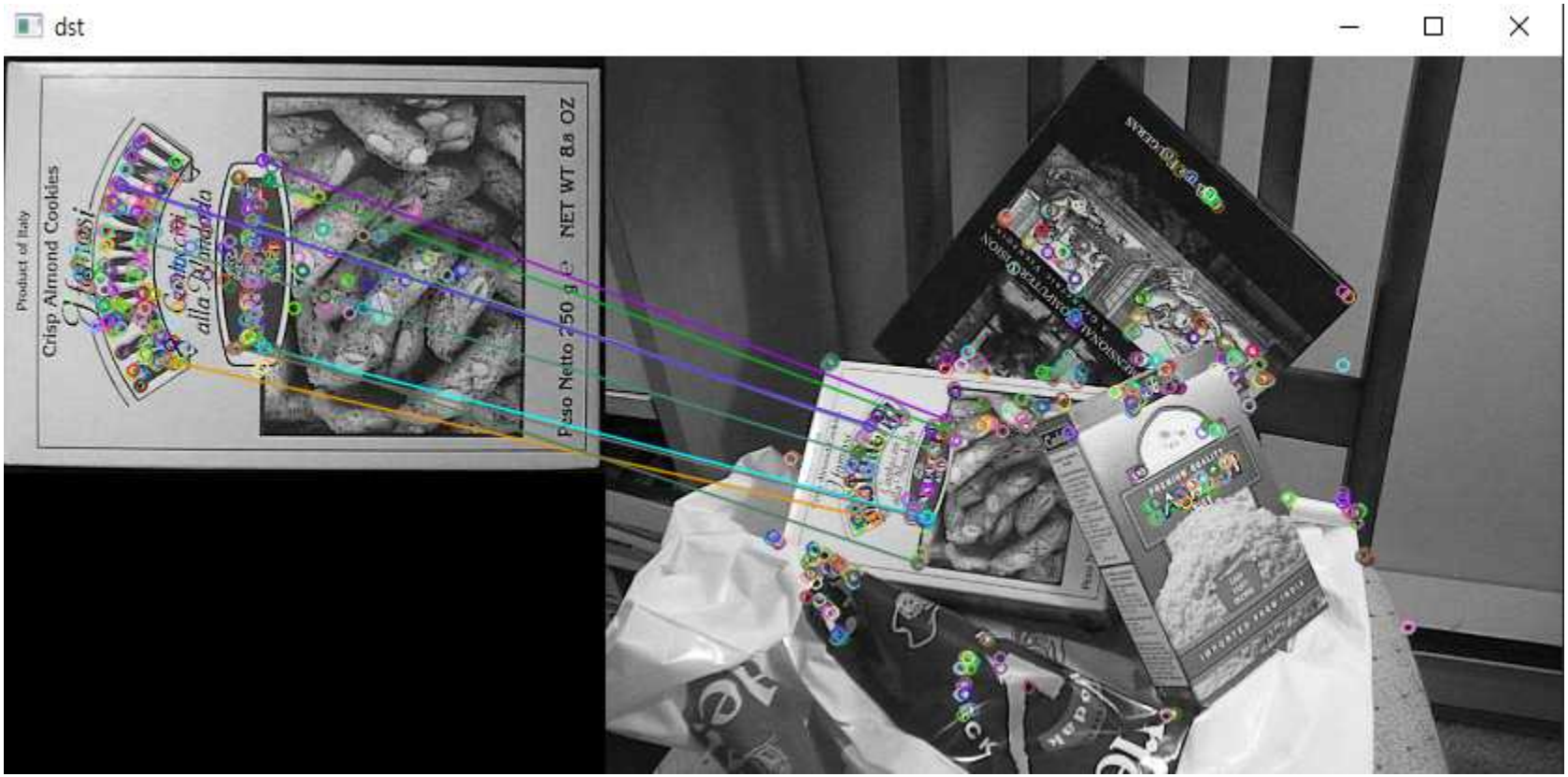
정렬 후 하위 50개의 matching만 추출한 결과물



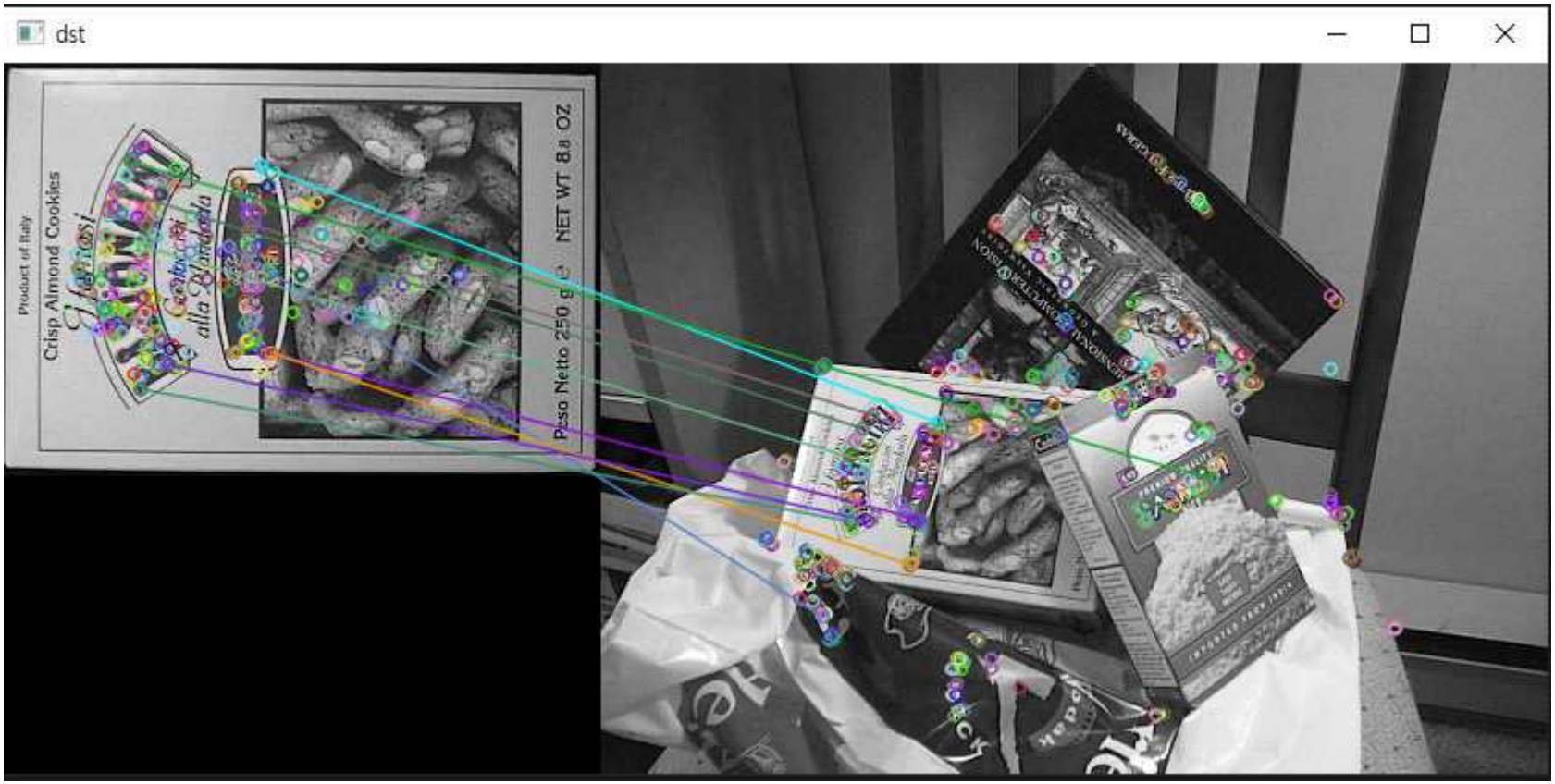
정렬 후 하위 10개의 matching만 추출한 결과물
상위matching보다 정확하지 않은 match임을 알 수 있다.



상위 50개 crossCheck를 true로 바꾼 결과물



상위 10개 BFMatcher의 기술자거리측정방식을 NORM_L1로 변경



상위 10개 BFMatcher의 기술자거리측정방식을 NORM_L2로 변경