

知能情報開発基礎セミナー最終レポート

中京大学大学院 工学研究科 機械システム工学専攻
T11703M 市川 智裕

目次

① OpenCV を使うパッケージの作成方法	・・・・・・・・ p.2
② USB カメラを使えるようにする方法	・・・・・・・・ p.5
③ 実行方法	・・・・・・・・ p.6
④ 作成したプログラムの概要	・・・・・・・・ p.8

① OpenCV を使うパッケージの作成方法

1. フォルダとその下に src ファイルをつくる.

```
$ mkdir -p my_program/src
```

2. my_program/src の中にパッケージをつくる.

```
$ cd my_program/src
```

```
$ catkin_create_pkg my_opencv sensor_msgs cv_bridge roscpp std_msgs image_transport
```

3. my_opencv/src の中ソースフォルダをつくる.

```
$ cd my_opencv/src
```

```
$ gedit optical_flow.cpp
```

4. プログラムを書く

プログラムの内容は4章を参照

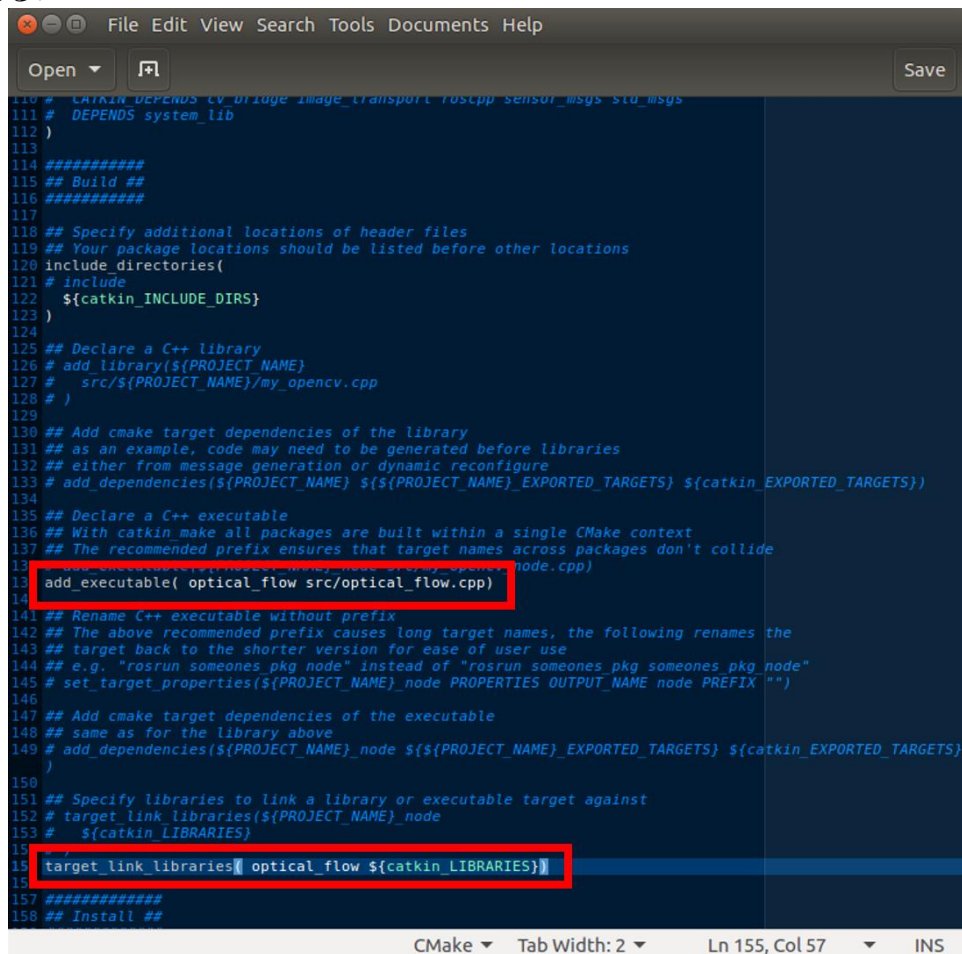
5. CMakeList.txt の編集

```
$ cd ..
```

```
$ gedit CMakeList.txt
```

139行目付近に `add_executable(optical_flow src/optical_flow.cpp)`

155行目付近に `target_link_libraries(optical_flow ${catkin_LIBRARIES})`
を加える.



```
File Edit View Search Tools Documents Help
Open [icon] Save
110 # catkin_DEPENDS cv_bridge image_transport roscpp sensor_msgs std_msgs
111 #   DEPENDS system_lib
112 )
113
114 #####
115 ## Build ##
116 #####
117
118 ## Specify additional locations of header files
119 ## Your package locations should be listed before other locations
120 include_directories(
121   # include
122   ${catkin_INCLUDE_DIRS}
123 )
124
125 ## Declare a C++ library
126 # add_library(${PROJECT_NAME}
127 #   src/${PROJECT_NAME}/my_opencv.cpp
128 # )
129
130 ## Add cmake target dependencies of the library
131 ## as an example, code may need to be generated before libraries
132 ## either from message generation or dynamic reconfigure
133 # add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
134
135 ## Declare a C++ executable
136 ## With catkin_make all packages are built within a single CMake context
137 ## The recommended prefix ensures that target names across packages don't collide
138 # add_executable(${PROJECT_NAME}_node src/${PROJECT_NAME}_node.cpp)
139 add_executable( optical_flow src/optical_flow.cpp)
140
141 ## Rename C++ executable without prefix
142 ## The above recommended prefix causes long target names, the following renames the
143 ## target back to the shorter version for ease of user use
144 ## e.g. "roslaunch someones_pkg node" instead of "roslaunch someones_pkg someones_pkg_node"
145 # set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFIX "")
146
147 ## Add cmake target dependencies of the executable
148 ## same as for the library above
149 # add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${catkin_EXPORTED_TARGETS})
150
151 ## Specify libraries to link a library or executable target against
152 # target_link_libraries(${PROJECT_NAME}_node
153 #   ${catkin_LIBRARIES}
154 # )
155 target_link_libraries( optical_flow ${catkin_LIBRARIES})
156
157 #####
158 ## Install ##
```

図 1. CMakeList.txt の編集箇所

6. CMakeList.txt をつくる.

もし my_program/src の中に CMakeList.txt があったら消す
my_program/src の中で

`$catkin_init_workspace`

のコマンドを打つと CMakeList.txt ができる.

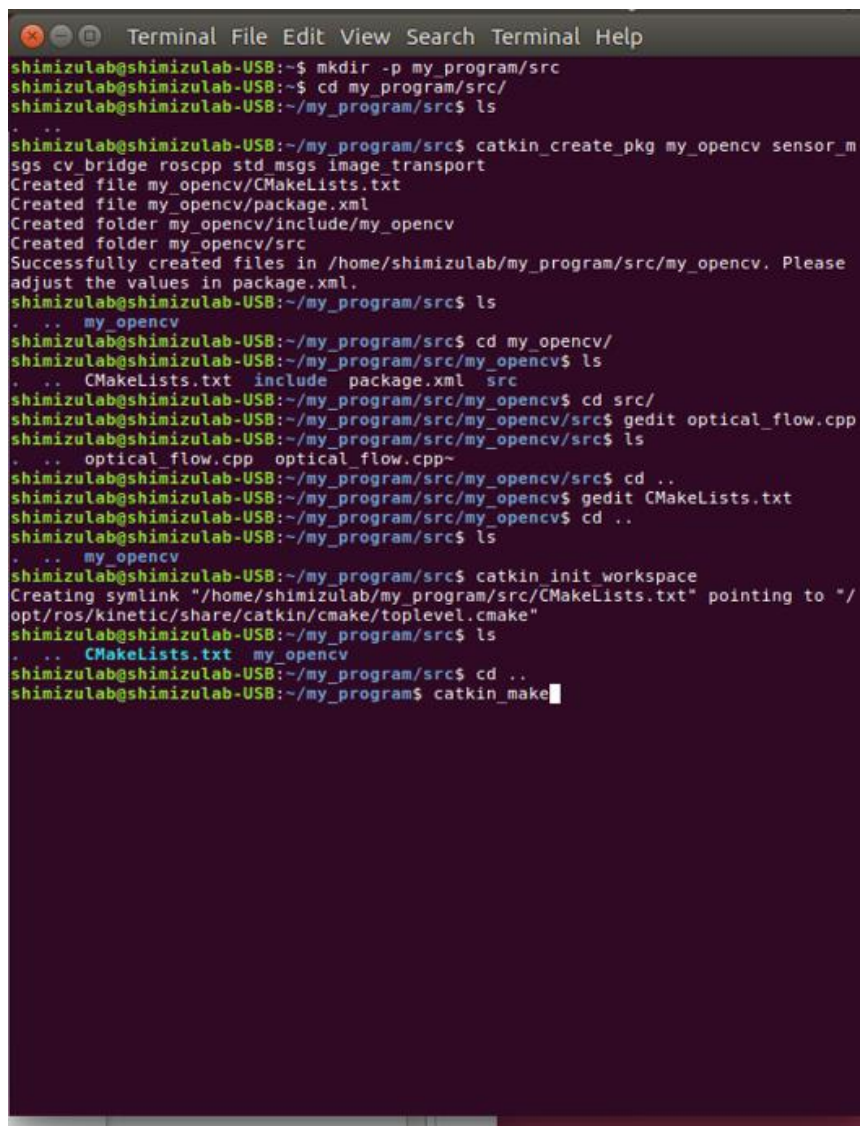
この中には、個々の PC のアドレスが書かれているため、他の PC で使う際は再度作り直す必要がある.

7. ビルドする.

my_program の階層で

`$catkin_make`

以上の流れで OpenCV を使ったプログラムのビルドを通すことができる.
下記にターミナルの実行結果を示す.



```
shimizulab@shimizulab-USB:~$ mkdir -p my_program/src
shimizulab@shimizulab-USB:~$ cd my_program/src/
shimizulab@shimizulab-USB:~/my_program/src$ ls
.
..
shimizulab@shimizulab-USB:~/my_program/src$ catkin_create_pkg my_opencv sensor_m
sgs cv_bridge roscpp std_msgs image_transport
Created file my_opencv/CMakeLists.txt
Created file my_opencv/package.xml
Created folder my_opencv/include/my_opencv
Created folder my_opencv/src
Successfully created files in /home/shimizulab/my_program/src/my_opencv. Please
adjust the values in package.xml.
shimizulab@shimizulab-USB:~/my_program/src$ ls
.
..
my_opencv
shimizulab@shimizulab-USB:~/my_program/src$ cd my_opencv/
shimizulab@shimizulab-USB:~/my_program/src/my_opencv$ ls
.
..
CMakeLists.txt include package.xml src
shimizulab@shimizulab-USB:~/my_program/src/my_opencv$ cd src/
shimizulab@shimizulab-USB:~/my_program/src/my_opencv/src$ gedit optical_flow.cpp
shimizulab@shimizulab-USB:~/my_program/src/my_opencv/src$ ls
.
..
optical_flow.cpp optical_flow.cpp~
shimizulab@shimizulab-USB:~/my_program/src/my_opencv/src$ cd ..
shimizulab@shimizulab-USB:~/my_program/src/my_opencv$ gedit CMakeLists.txt
shimizulab@shimizulab-USB:~/my_program/src/my_opencv$ cd ..
shimizulab@shimizulab-USB:~/my_program/src$ ls
.
..
my_opencv
shimizulab@shimizulab-USB:~/my_program/src$ catkin_init workspace
Creating symlink "/home/shimizulab/my_program/src/CMakeLists.txt" pointing to "/"
opt/ros/kinetic/share/catkin/cmake/toplevel.cmake"
shimizulab@shimizulab-USB:~/my_program/src$ ls
.
..
CMakeLists.txt my_opencv
shimizulab@shimizulab-USB:~/my_program/src$ cd ..
shimizulab@shimizulab-USB:~/my_program$ catkin_make
```

図 2. ターミナルへの記述内容 その 1

```
Terminal File Edit View Search Terminal Help
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Using CATKIN_DEVEL_PREFIX: /home/shimizulab/my_program/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/kinetic
-- This workspace overlays: /opt/ros/kinetic
-- Found PythonInterp: /usr/bin/python (found version "2.7.12")
-- Using PYTHON_EXECUTABLE: /usr/bin/python
-- Using Debian Python package layout
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/shimizulab/my_program/build/test_results
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Looking for pthread_create
-- Looking for pthread_create - not found
-- Looking for pthread_create in pthreads
-- Looking for pthread_create in pthreads - not found
-- Looking for pthread_create in pthread
-- Looking for pthread_create in pthread - found
-- Found Threads: TRUE
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.6
-- BUILD_SHARED_LIBS is on
~~~~~
-- ~~~ traversing 1 packages in topological order:
-- ~~~ - my_opencv
~~~~~
-- +++ processing catkin package: 'my_opencv'
-- ==> add_subdirectory(my_opencv)
-- Configuring done
-- Generating done
-- Build files have been written to: /home/shimizulab/my_program/build
####
#### Running command: "make -j2 -l2" in "/home/shimizulab/my_program/build"
####
Scanning dependencies of target optical_flow
[ 50%] Building CXX object my_opencv/CMakeFiles/optical_flow.dir/src/optical_flow.cpp.o
[100%] Linking CXX executable /home/shimizulab/my_program/devel/lib/my_opencv/optical_flow
[100%] Built target optical_flow
shimizulab@shimizulab-USB:~/my_program$
```

図 3. ターミナルへの記述内容 その 2

② USB カメラを使えるようにする方法

1. 使用するプログラムがある src ファイルに移動

```
$cd my_program/src
```

2. github のサイトから ROS で USB カメラを使えるソフトをインストールする.

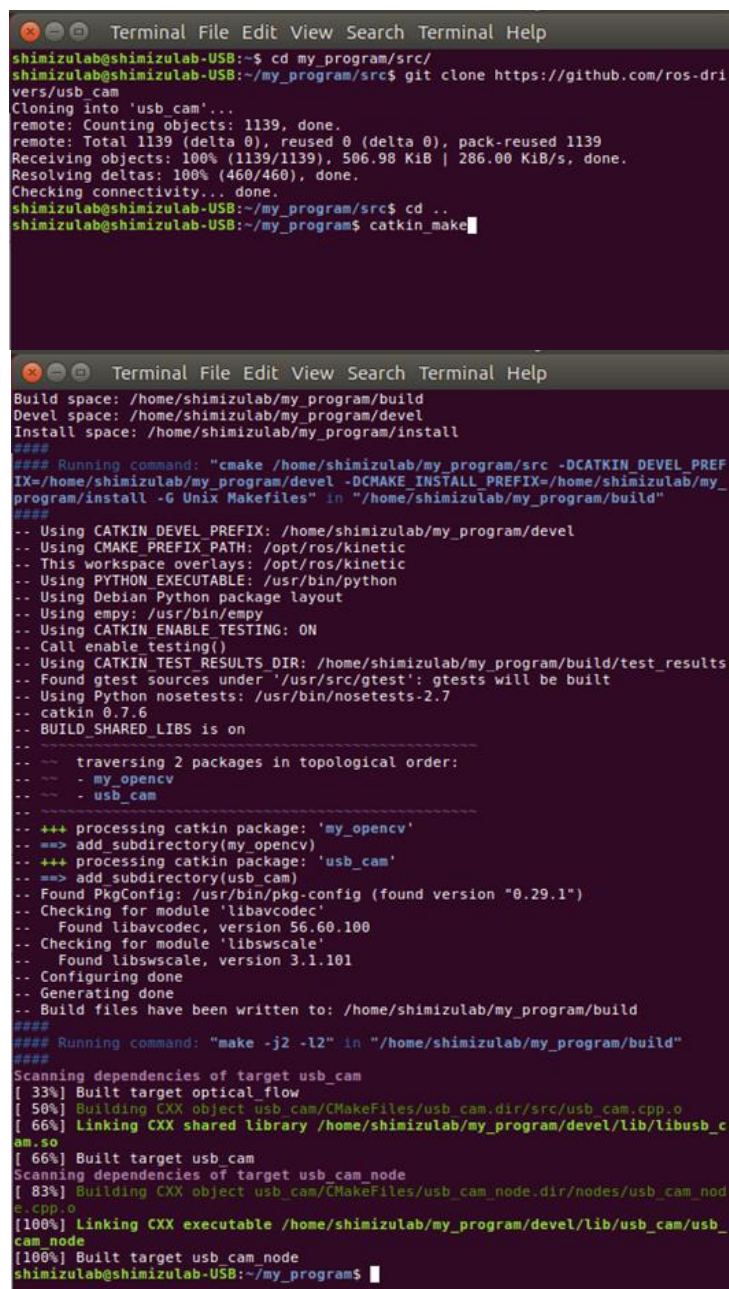
```
$git clone https://github.com/ros-drivers/usb_cam
```

3. ビルドする.

```
$cd ..
```

```
$catkin_make
```

以上の流れで USB カメラが使えるようになる。



```
Terminal File Edit View Search Terminal Help
shimizulab@shimizulab-USB:~$ cd my_program/src/
shimizulab@shimizulab-USB:~/my_program/src$ git clone https://github.com/ros-dri
vers/usb_cam
Cloning into 'usb_cam'...
remote: Counting objects: 1139, done.
remote: Total 1139 (delta 0), reused 0 (delta 0), pack-reused 1139
Receiving objects: 100% (1139/1139), 506.98 KiB | 286.00 KiB/s, done.
Resolving deltas: 100% (460/460), done.
Checking connectivity... done.
shimizulab@shimizulab-USB:~/my_program/src$ cd ..
shimizulab@shimizulab-USB:~/my_program$ catkin_make

Build space: /home/shimizulab/my_program/build
Devel space: /home/shimizulab/my_program/devel
Install space: /home/shimizulab/my_program/install
####
### Running command: "cmake /home/shimizulab/my_program/src -DCATKIN_DEVEL_PREF
IX=/home/shimizulab/my_program/devel -DCMAKE_INSTALL_PREFIX=/home/shimizulab/my_
program/install -G Unix Makefiles" in "/home/shimizulab/my_program/build"
####
-- Using CATKIN DEVEL_PREFIX: /home/shimizulab/my_program/devel
-- Using CMAKE_PREFIX_PATH: /opt/ros/kinetic
-- This workspace overlays: /opt/ros/kinetic
-- Using PYTHON_EXECUTABLE: /usr/bin/python
-- Using Debian Python package layout
-- Using empy: /usr/bin/empy
-- Using CATKIN_ENABLE_TESTING: ON
-- Call enable_testing()
-- Using CATKIN_TEST_RESULTS_DIR: /home/shimizulab/my_program/build/test_results
-- Found gtest sources under '/usr/src/gtest': gtests will be built
-- Using Python nosetests: /usr/bin/nosetests-2.7
-- catkin 0.7.6
-- BUILD_SHARED_LIBS is on
--
-- traversing 2 packages in topological order:
-- - my_opencv
-- - usb_cam
--
-- ++ processing catkin package: 'my_opencv'
-- ==> add_subdirectory(my_opencv)
-- ++ processing catkin package: 'usb_cam'
-- ==> add_subdirectory(usb_cam)
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.1")
-- Checking for module 'libavcodec'
-- Found libavcodec, version 56.60.100
-- Checking for module 'libswscale'
-- Found libswscale, version 3.1.101
-- Configuring done
-- Generating done
-- Build files have been written to: /home/shimizulab/my_program/build
####
### Running command: "make -j2 -l2" in "/home/shimizulab/my_program/build"
####
Scanning dependencies of target usb_cam
[ 33%] Built target optical_flow
[ 50%] Building CXX object usb_cam/CMakeFiles/usb_cam.dir/src/usb_cam.cpp.o
[ 66%] Linking CXX shared library /home/shimizulab/my_program/devel/lib/libusb_c
am.so
[ 66%] Built target usb_cam
Scanning dependencies of target usb_cam_node
[ 83%] Building CXX object usb_cam/CMakeFiles/usb_cam_node.dir/nodes/usb_cam_nod
e.cpp.o
[100%] Linking CXX executable /home/shimizulab/my_program/devel/lib/usb_cam/usb_
cam_node
[100%] Built target usb_cam_node
shimizulab@shimizulab-USB:~/my_program$
```

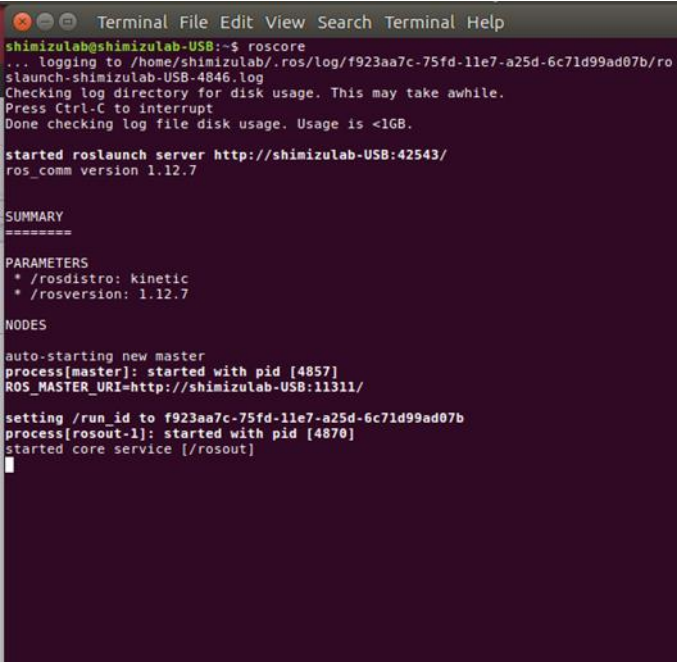
図 3. ターミナルへの記述内容 その 3

③ OpenCV と USB カメラ，ROS を用いたプログラムの実行方法

実行には3つのターミナルを用いる．

ターミナル1：ROS を動かす

```
$roscore
```

A screenshot of a terminal window titled "Terminal File Edit View Search Terminal Help". The terminal shows the output of the command "roscore" in a dark-themed environment. The output includes logging information, a check for disk usage, and the successful launch of the ROS master and core service. The terminal text is as follows:

```
shimizulab@shimizulab-USB:~$ roscore
... logging to /home/shimizulab/.ros/log/f923aa7c-75fd-11e7-a25d-6c71d99ad07b/ro
slaunch-shimizulab-USB-4846.log
Checking log directory for disk usage. This may take awhile.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.

started roslaunch server http://shimizulab-USB:42543/
ros_comm version 1.12.7

SUMMARY
=====
PARAMETERS
* /rostdistro: kinetic
* /rosversion: 1.12.7

NODES

auto-starting new master
process[master]: started with pid [4857]
ROS_MASTER_URI=http://shimizulab-USB:11311/

setting /run_id to f923aa7c-75fd-11e7-a25d-6c71d99ad07b
process[rosout-1]: started with pid [4870]
started core service [/rosout]
```

図4．ターミナル1への記述内容

ターミナル2：USB カメラを使えるようにする．

```
$cd my_program
```

```
$source devel/setup.bash
```

(\$rosparm set usb_cam/pixel_format yuyv)カメラによってはフォーマットを指定する必要がある．

```
$roslaunch usb_cam usb_cam_node
```

```
Terminal File Edit View Search Terminal Tabs Help
roscore http://shimizulab-USB... x shimizulab@shimizulab-USB: ~... x +
shimizulab@shimizulab-USB:~$ cd my_program/
shimizulab@shimizulab-USB:~/my_program$ source devel/setup.bash
shimizulab@shimizulab-USB:~/my_program$ rosparam set usb_cam/pixel_format yuyv
shimizulab@shimizulab-USB:~/my_program$ roslaunch usb_cam usb_cam_node
[ INFO] [1501512028.309547829]: using default calibration URL
[ INFO] [1501512028.309904880]: camera calibration URL: file:///home/shimizulab/
.ros/camera_info/head_camera.yaml
[ INFO] [1501512028.310129690]: Unable to open camera calibration file [/home/sh
imizulab/.ros/camera_info/head_camera.yaml]
[ WARN] [1501512028.310280453]: Camera calibration file /home/shimizulab/.ros/ca
mera_info/head_camera.yaml not found.
[ INFO] [1501512028.310435737]: Starting 'head_camera' (/dev/video0) at 640x480
via mmap (yuyv) at 30 FPS
[ WARN] [1501512028.716402134]: sh: 1: v4l2-ctl: not found
[ WARN] [1501512028.721043955]: sh: 1: v4l2-ctl: not found
```

図 5. ターミナル 2 への記述内容

ターミナル 3 : OpenCV で動かす
\$cd my_program
\$source devel/setup.bash
\$roslaunch my_opencv optical_flow

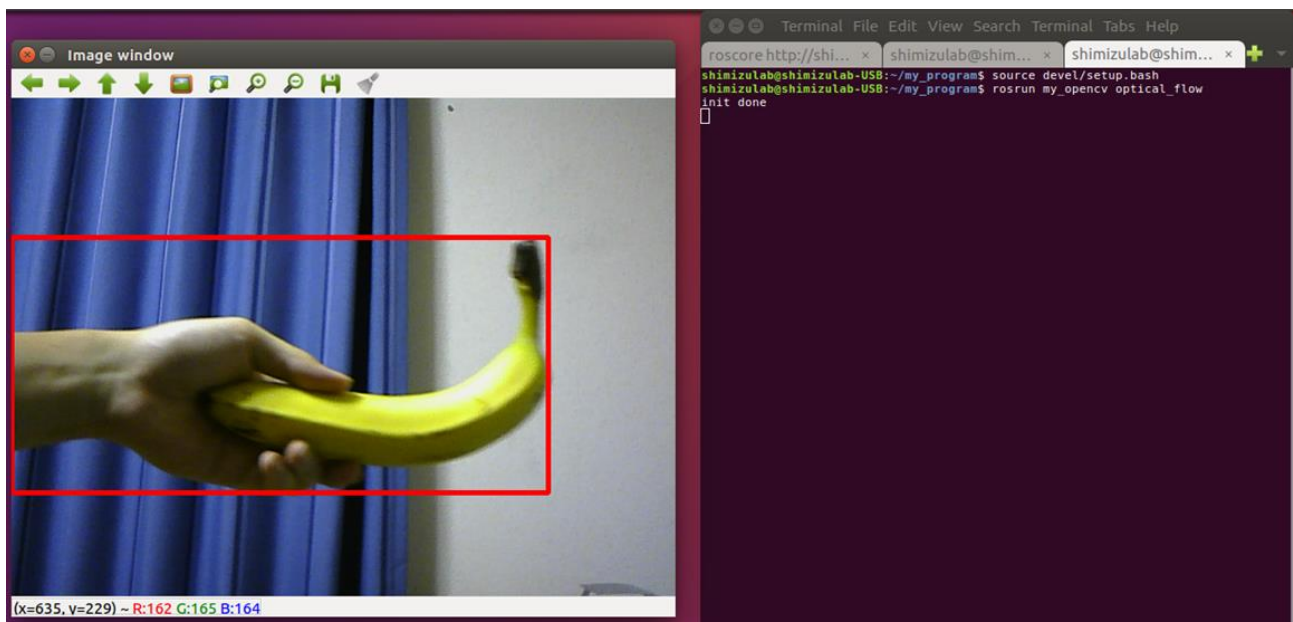
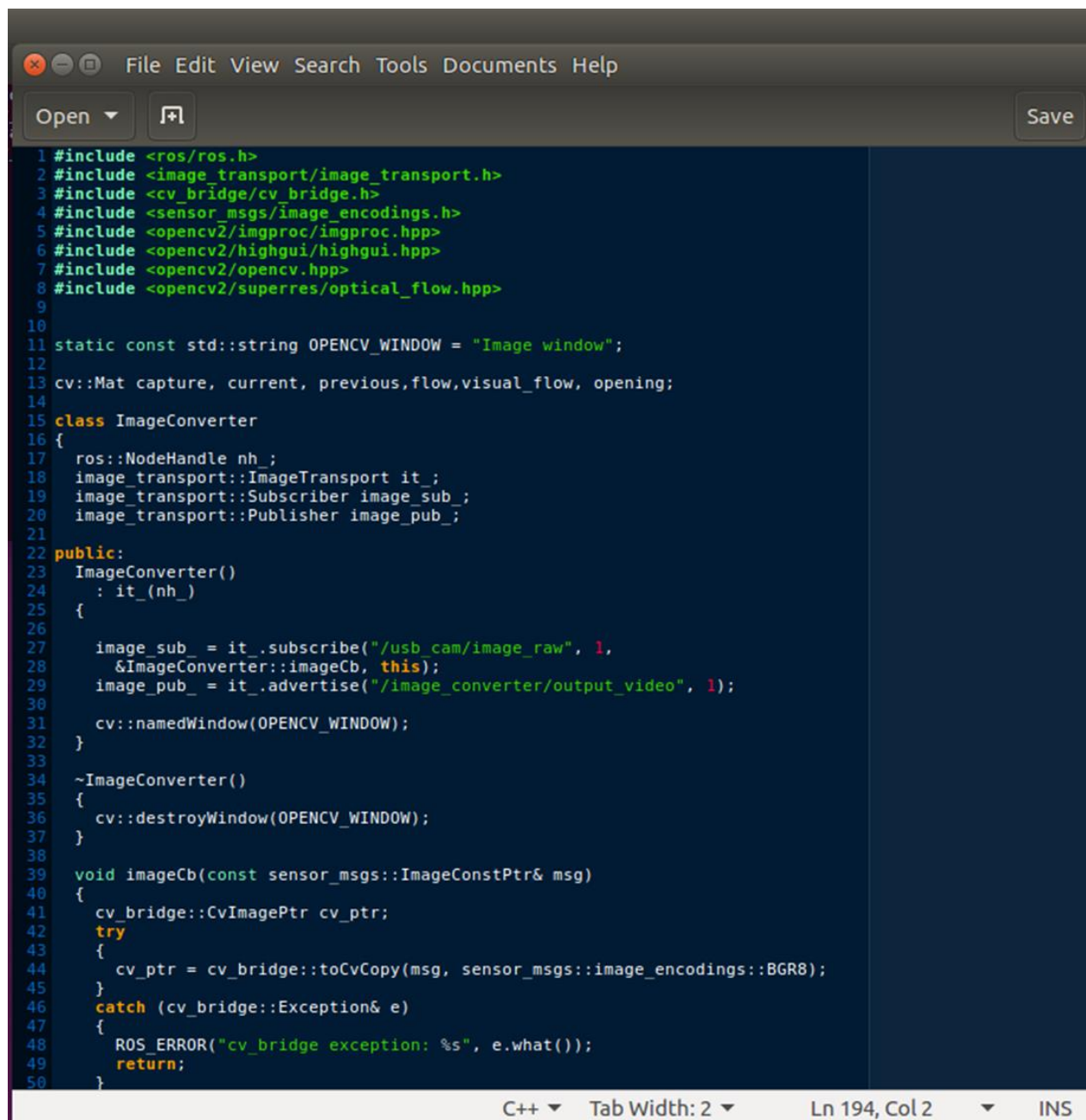


図 6. ターミナル 3 への記述内容と実行結果

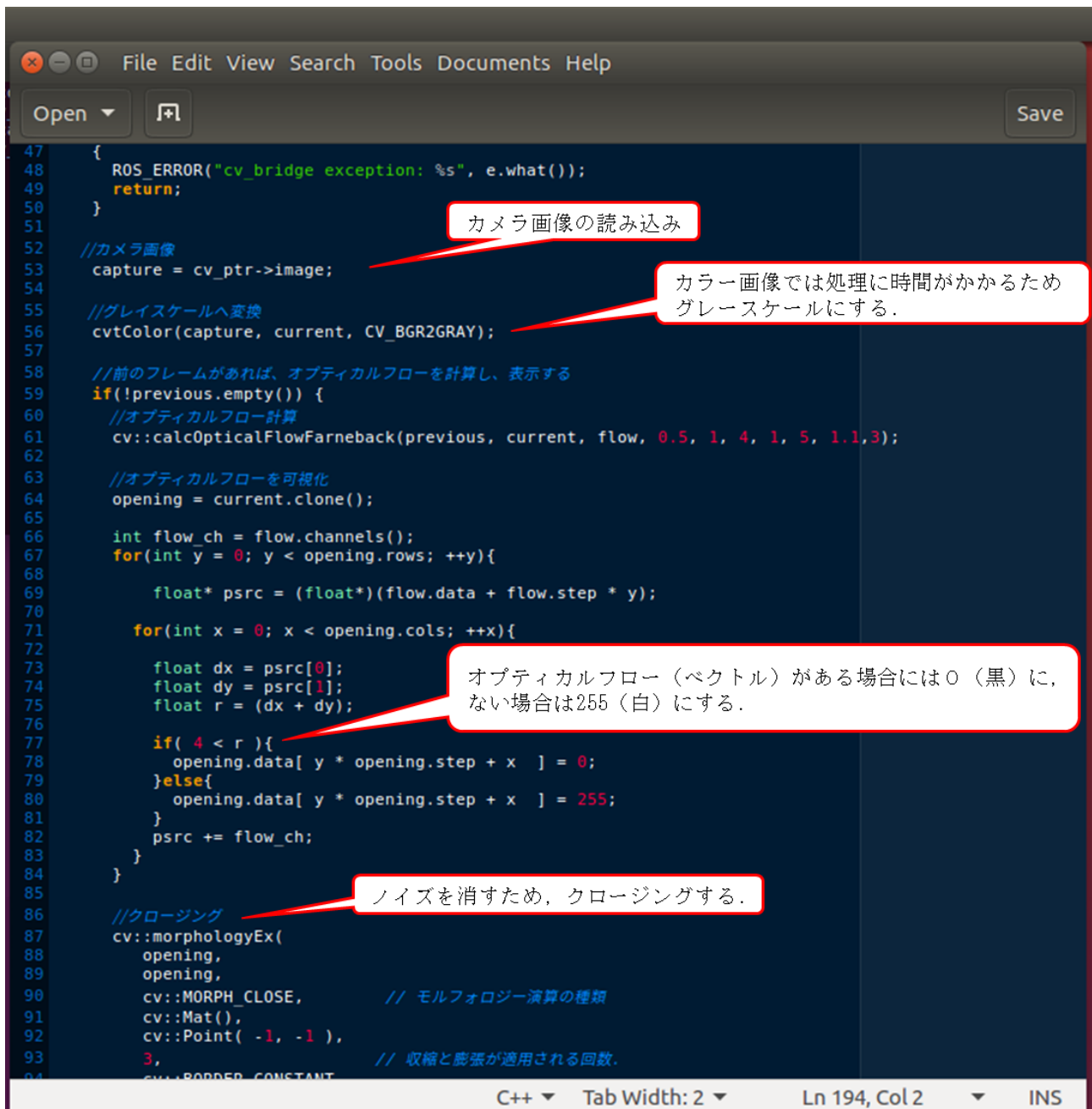
④ プログラムの概要

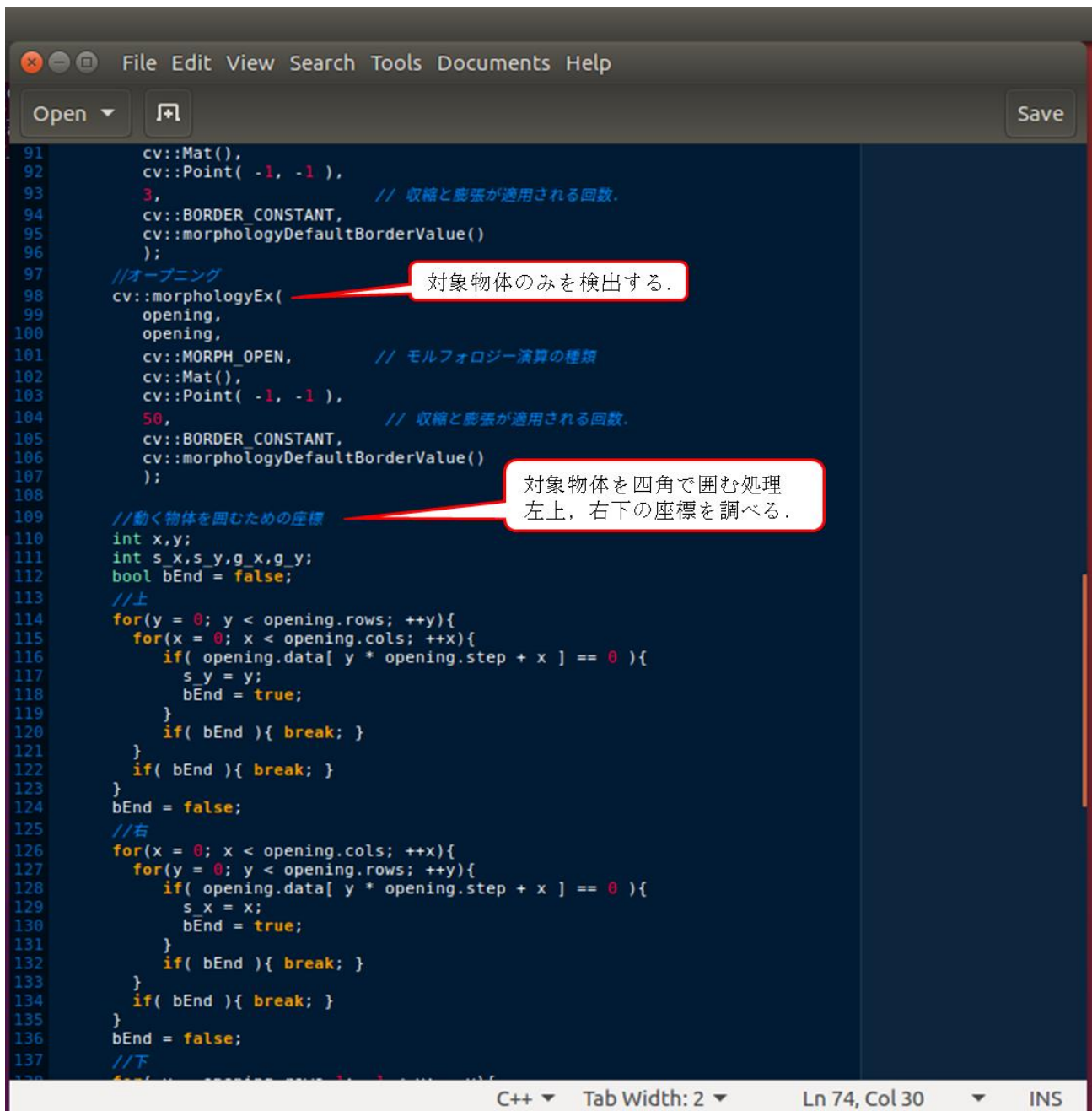
授業で扱った，カメラ画像を表示するプログラムを元に作成した．



```
1 #include <ros/ros.h>
2 #include <image_transport/image_transport.h>
3 #include <cv_bridge/cv_bridge.h>
4 #include <sensor_msgs/image_encodings.h>
5 #include <opencv2/imgproc/imgproc.hpp>
6 #include <opencv2/highgui/highgui.hpp>
7 #include <opencv2/opencv.hpp>
8 #include <opencv2/superres/optical_flow.hpp>
9
10 static const std::string OPENCV_WINDOW = "Image window";
11 cv::Mat capture, current, previous, flow, visual_flow, opening;
12
13 class ImageConverter
14 {
15     ros::NodeHandle nh_;
16     image_transport::ImageTransport it_;
17     image_transport::Subscriber image_sub_;
18     image_transport::Publisher image_pub_;
19
20 public:
21     ImageConverter()
22     : it_(nh_)
23     {
24         image_sub_ = it_.subscribe("/usb_cam/image_raw", 1,
25             &ImageConverter::imageCb, this);
26         image_pub_ = it_.advertise("/image_converter/output_video", 1);
27
28         cv::namedWindow(OPENCV_WINDOW);
29     }
30
31     ~ImageConverter()
32     {
33         cv::destroyWindow(OPENCV_WINDOW);
34     }
35
36     void imageCb(const sensor_msgs::ImageConstPtr& msg)
37     {
38         cv_bridge::CvImagePtr cv_ptr;
39         try
40         {
41             cv_ptr = cv_bridge::toCvCopy(msg, sensor_msgs::image_encodings::BGR8);
42         }
43         catch (cv_bridge::Exception& e)
44         {
45             ROS_ERROR("cv_bridge exception: %s", e.what());
46             return;
47         }
48     }
49 }
```

C++ Tab Width: 2 Ln 194, Col 2 INS





```
91 cv::Mat(),
92 cv::Point( -1, -1 ),
93 3, // 収縮と膨張が適用される回数.
94 cv::BORDER_CONSTANT,
95 cv::morphologyDefaultBorderValue()
96 );
97 //オープニング
98 cv::morphologyEx( // 対象物体のみを検出する.
99 opening,
100 opening,
101 cv::MORPH_OPEN, // モルフォロジー演算の種類
102 cv::Mat(),
103 cv::Point( -1, -1 ),
104 50, // 収縮と膨張が適用される回数.
105 cv::BORDER_CONSTANT,
106 cv::morphologyDefaultBorderValue()
107 );
108
109 //動く物体を囲むための座標
110 int x,y;
111 int s_x,s_y,g_x,g_y;
112 bool bEnd = false;
113 //上
114 for(y = 0; y < opening.rows; ++y){
115     for(x = 0; x < opening.cols; ++x){
116         if( opening.data[ y * opening.step + x ] == 0 ){
117             s_y = y;
118             bEnd = true;
119         }
120         if( bEnd ){ break; }
121     }
122     if( bEnd ){ break; }
123 }
124 bEnd = false;
125 //右
126 for(x = 0; x < opening.cols; ++x){
127     for(y = 0; y < opening.rows; ++y){
128         if( opening.data[ y * opening.step + x ] == 0 ){
129             s_x = x;
130             bEnd = true;
131         }
132         if( bEnd ){ break; }
133     }
134     if( bEnd ){ break; }
135 }
136 bEnd = false;
137 //下
138 for(y = opening.rows-1; y >= 0; --y){
139     for(x = 0; x < opening.cols; ++x){
140         if( opening.data[ y * opening.step + x ] == 0 ){
141             g_y = y;
142             bEnd = true;
143         }
144         if( bEnd ){ break; }
145     }
146     if( bEnd ){ break; }
147 }
148 bEnd = false;
149 //左
150 for(x = opening.cols-1; x >= 0; --x){
151     for(y = 0; y < opening.rows; ++y){
152         if( opening.data[ y * opening.step + x ] == 0 ){
153             g_x = x;
154             bEnd = true;
155         }
156         if( bEnd ){ break; }
157     }
158     if( bEnd ){ break; }
159 }
160 bEnd = false;
```

C++ Tab Width: 2 Ln 74, Col 30 INS

```
File Edit View Search Tools Documents Help
Open Save

130     bEnd = true;
131     }
132     if( bEnd ){ break; }
133     }
134     if( bEnd ){ break; }
135     }
136     bEnd = false;
137     //下
138     for( y = opening.rows-1; -1 < y; --y){
139         for( x = opening.cols-1; -1 < x; --x){
140             if( opening.data[ y * opening.step + x ] == 0 ){
141                 g_y = y;
142                 bEnd = true;
143             }
144             if( bEnd ){ break; }
145         }
146         if( bEnd ){ break; }
147     }
148     bEnd = false;
149     //左
150     for( x = opening.cols-1; -1 < x; --x){
151         for( y = opening.rows-1; -1 < y; --y){
152             if( opening.data[ y * opening.step + x ] == 0 ){
153                 g_x = x;
154                 bEnd = true;
155             }
156             if( bEnd ){ break; }
157         }
158         if( bEnd ){ break; }
159     }
160     bEnd = false;
161
162
163     //表示画面
164     visual_flow = capture.clone();
165
166     //動く物体を四角で囲む
167     cv::rectangle(visual_flow, cv::Point(s_x,s_y), cv::Point(g_x, g_y), cv::Scalar(0,0,255), 3, 4);
168
169     //表示
170     cv::imshow(OPENCV_WINDOW, visual_flow);
171     //printf("OK\n");
172 }
173 //前のフレームを保存
174 previous = current.clone();
175
176
177 cv::waitKey(3);
```

```
File Edit View Search Tools Documents Help
Open Save

147 }
148 bEnd = false;
149 //左
150 for( x = opening.cols-1; -1 < x; --x){
151     for( y = opening.rows-1; -1 < y; --y){
152         if( opening.data[ y * opening.step + x ] == 0 ){
153             g_x = x;
154             bEnd = true;
155         }
156         if( bEnd ){ break; }
157     }
158     if( bEnd ){ break; }
159 }
160 bEnd = false;
161
162
163 //表示画面
164 visual_flow = capture.clone();
165
166 //動く物体を四角で囲む
167 cv::rectangle(visual_flow, cv::Point(s_x,s_y), cv::Point(g_x, g_y), cv::Scalar(0,0,255), 3, 4);
168
169 //表示
170 cv::imshow(OPENCV_WINDOW, visual_flow);
171 //printf("OK\n");
172 }
173 //前のフレームを保存
174 previous = current.clone();
175
176
177 cv::waitKey(3);
178
179 // Output modified video stream
180 image_pub_.publish(cv_ptr->toImageMsg());
181
182 }
183 };
184
185
186 int main(int argc, char** argv)
187 {
188
189     ros::init(argc, argv, "image_converter");
190     ImageConverter ic;
191     ros::spin();
192     return 0;
193 }
194 }
```

C++ Tab Width: 2 Ln 118, Col 26 INS