無人載具技術與應用
ROS Computer Vision

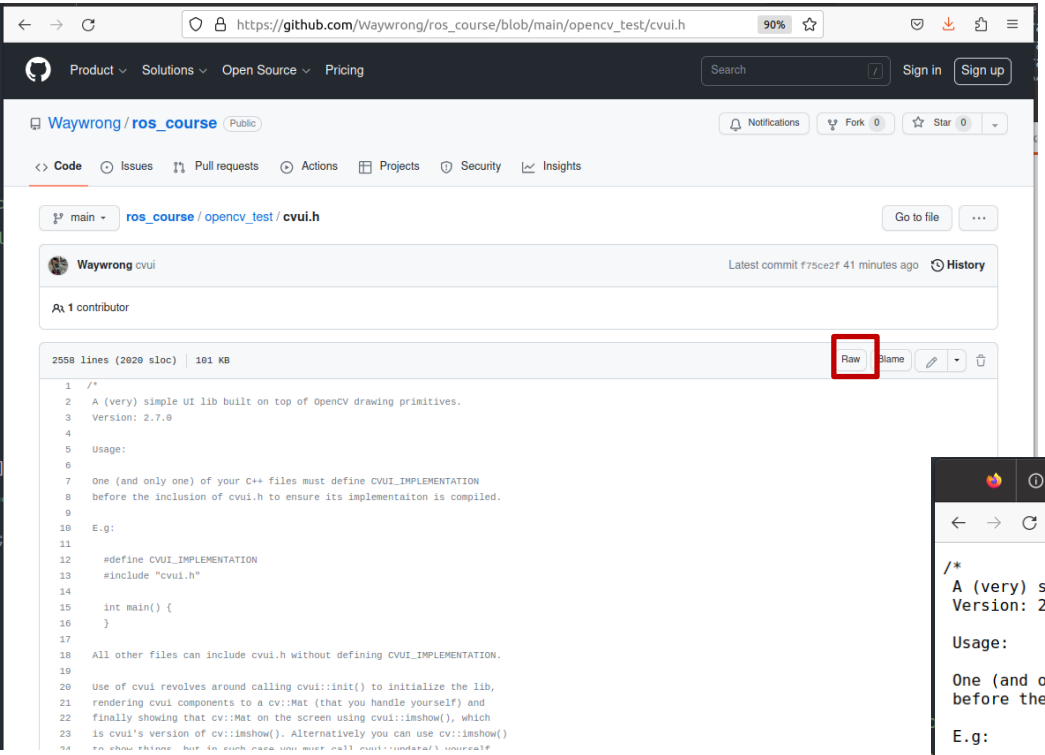
徐瑋隆
wlhsu304@gmail.com

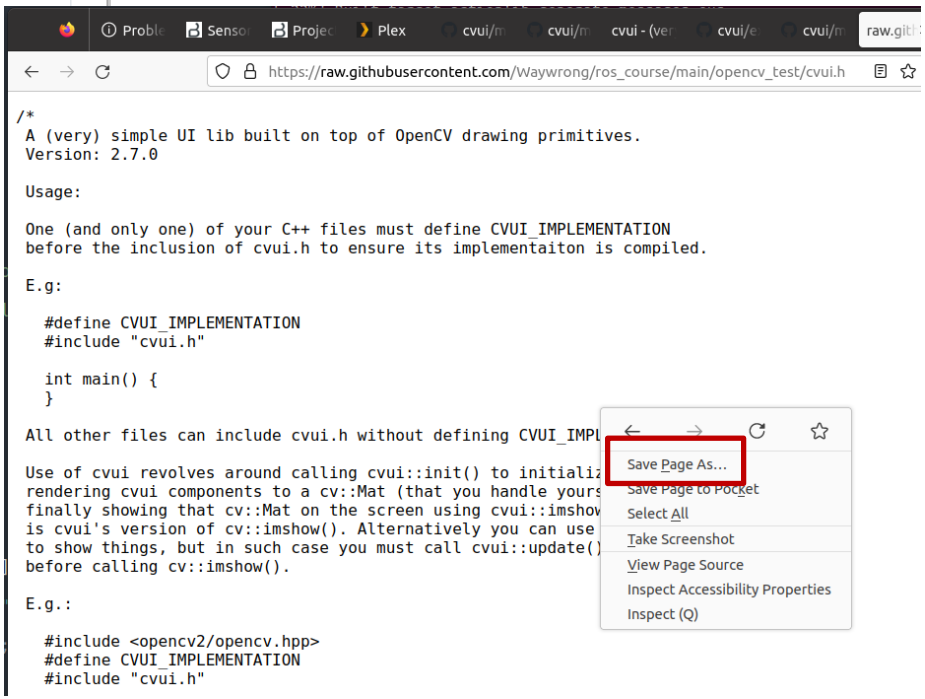
OpenCV UI 01

ROS COMPUTER VISION

cv 01

https://github.com/Waywrong/ros_course/blob/main/opencv_test





cp ~/Downloads/cvui.h ~/projects/opencv/

cv 02

```
user@user-virtual-machine:~$ cd projects/opencv/
user@user-virtual-machine:~/projects/opencv$ gedit cv_ui01.cpp &
```

Open | cv_ui01.cpp ~/projects/opencv | Save

```cpp
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#define WINDOW_NAME "Button"

int main(int argc, const char *argv[])
{
  cv::Mat frame = cv::Mat(300, 600, CV_8UC3);
  cvui::init(WINDOW_NAME);
  std::string btn_name = "OFF";
  bool bBtn = false;
  while (true) {
    frame = cv::Scalar(49, 52, 49);
    cvui::text(frame, 150, 50, "Button example");
    if (cvui::button(frame, 360, 80, btn_name)) {
      std::cout << "Regular button clicked!" << std::endl;
      bBtn=!bBtn;
      btn_name = bBtn? "ON":"OFF";
    }
    cvui::update();
    cv::imshow(WINDOW_NAME, frame);
    // Check if ESC key was pressed
    if (cv::waitKey(20) == 27) {
      break;
    }
  }
  return 0;
}
```

```
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#define WINDOW_NAME "Button"

int main(int argc, const char *argv[])
{
  cv::Mat frame = cv::Mat(300, 600, CV_8UC3);
  cvui::init(WINDOW_NAME);
  std::string btn_name = "OFF";
  bool bBtn = false;
  while (true) {
    frame = cv::Scalar(49, 52, 49);
    cvui::text(frame, 150, 50, "Button example");
    if (cvui::button(frame, 360, 80, btn_name)) {
      std::cout << "Regular button clicked!" << std::endl;
      bBtn=!bBtn;
      btn_name = bBtn? "ON":"OFF";
    }
    cvui::update();
    cv::imshow(WINDOW_NAME, frame);
    // Check if ESC key was pressed
    if (cv::waitKey(20) == 27) {
      break;
    }
  }
  return 0;
}
```

cv 03

```
user@user-virtual-machine:~/projects/opencv$ gedit CMakeLists.txt &
```

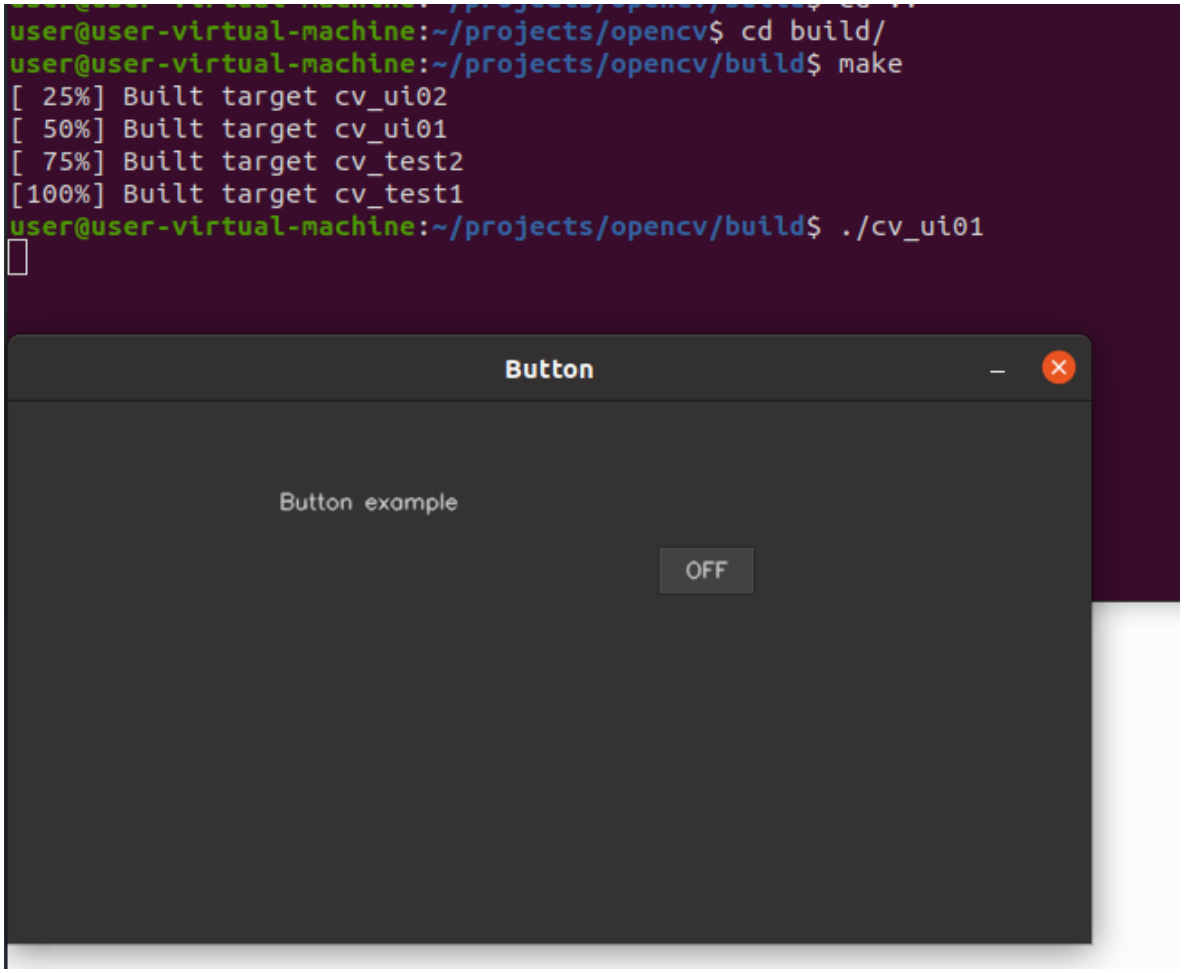
Open | CMakeLists.txt ~/projects/opencv | Save

cv_ui01.cpp | CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
project( DisplayImage )
find_package( OpenCV REQUIRED )

add_executable( cv_test1 cv_test1.cpp )
target_link_libraries( cv_test1 ${OpenCV_LIBS} )

add_executable( cv_test2 cv_test2.cpp )
target_link_libraries( cv_test2 ${OpenCV_LIBS} )

add_executable( cv_ui01 cv_ui01.cpp )
target_link_libraries( cv_ui01 ${OpenCV_LIBS} )
```

add_executable(cv_ui01 cv_ui01.cpp)
target_link_libraries(cv_ui01 ${OpenCV_LIBS})

cv 04


user@user-virtual-machine:~/projects/opencv$ cd build/
user@user-virtual-machine:~/projects/opencv/build$ make
[25%] Built target cv_ui02
[50%] Built target cv_ui01
[75%] Built target cv_test2
[100%] Built target cv_test1
user@user-virtual-machine:~/projects/opencv/build$ ./cv_ui01
Button
Button example
OFF

OpenCV UI 02

ROS COMPUTER VISION

cv 01

```
ser@user-virtual-machine:~/projects/opencv/build$ cd ..
ser@user-virtual-machine:~/projects/opencv$ gedit cv_ui02.cpp
ser@user-virtual-machine:~/projects/opencv$
```

```cpp
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

#define CVUI_IMPLEMENTATION
#include "cvui.h"

#define WINDOW_NAME "Trackbar and columns"

int main(int argc, const char *argv[])
{
  int intValue1 = 30;
  float floatValue1 = 12.;
  cv::Mat frame = cv::Mat(cv::Size(450, 250), CV_8UC3);

  // Size of trackbars
  int width = 400;

  cvui::init(WINDOW_NAME, 20);

  while (true) {
    frame = cv::Scalar(49, 52, 49);

    cvui::beginColumn(frame, 20, 20, -1, -1, 6);
    cvui::text("int trackbar, no customization");
    cvui::trackbar(width, &intValue1, 0, 100);
    cvui::space(5);

    cvui::text("float trackbar, no customization");
    cvui::trackbar(width, &floatValue1, 10.f, 15.f);
    cvui::space(5);

    std::cout<<"intValue1:"<<intValue1<<std::endl;
    std::cout<<"floatValue1:"<<floatValue1<<std::endl;
    if (cvui::button("&Quit")) {
      break;
    }
    cvui::endColumn();

    cvui::update();

    cv::imshow(WINDOW_NAME, frame);
  }

  return 0;
}
```

```
#include <iostream>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>

#define CVUI_IMPLEMENTATION
#include "cvui.h"

#define WINDOW_NAME "Trackbar and columns"

int main(int argc, const char *argv[])
{
  int intValue1 = 30;
  float floatValue1 = 12.;
  cv::Mat frame = cv::Mat(cv::Size(450, 250), CV_8UC3);

  // Size of trackbars
  int width = 400;

  cvui::init(WINDOW_NAME, 20);

  while (true) {
    frame = cv::Scalar(49, 52, 49);

    cvui::beginColumn(frame, 20, 20, -1, -1, 6);
    cvui::text("int trackbar, no customization");
    cvui::trackbar(width, &intValue1, 0, 100);
    cvui::space(5);

    cvui::text("float trackbar, no customization");
    cvui::trackbar(width, &floatValue1, 10.f, 15.f);
    cvui::space(5);

    std::cout<<"intValue1:"<<intValue1<<std::endl;
    std::cout<<"floatValue1:"<<floatValue1<<std::endl;
    if (cvui::button("&Quit")) {
      break;
    }
    cvui::endColumn();

    cvui::update();

    cv::imshow(WINDOW_NAME, frame);
  }

  return 0;
}
```

cv 02

```
user@user-virtual-machine:~/projects/opencv$ gedit CMakeLists.txt &
```

Open

CMakeLists.txt
~/projects/opencv

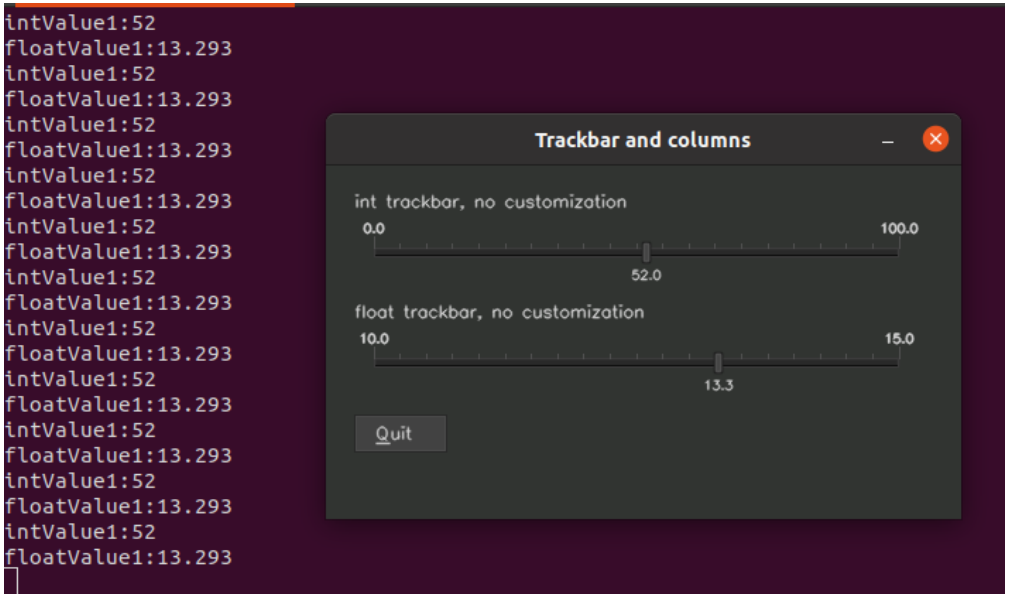
cv_ui01.cpp | CMakeLists.txt

```
1 cmake_minimum_required(VERSION 2.8)
2 project( DisplayImage )
3 find_package( OpenCV REQUIRED )
4
5 add_executable( cv_test1 cv_test1.cpp )
6 target_link_libraries( cv_test1 ${OpenCV_LIBS} )
7
8 add_executable( cv_test2 cv_test2.cpp )
9 target_link_libraries( cv_test2 ${OpenCV_LIBS} )
10
11 add_executable( cv_ui01 cv_ui01.cpp )
12 target_link_libraries( cv_ui01 ${OpenCV_LIBS} )
13
14 add_executable( cv_ui02 cv_ui02.cpp )
15 target_link_libraries( cv_ui02 ${OpenCV_LIBS} )
```

add_executable(cv_ui02 cv_ui02.cpp)
target_link_libraries(cv_ui02 ${OpenCV_LIBS})

cv 03

```
user@user-virtual-machine:~/projects/opencv$ cd build/
user@user-virtual-machine:~/projects/opencv/build$ make
[ 25%] Built target cv_ui02
[ 50%] Built target cv_ui01
[ 75%] Built target cv_test2
[100%] Built target cv_test1
user@user-virtual-machine:~/projects/opencv/build$ ./cv_ui02
```



ROS + OpenCV UI : Topic

ROS COMPUTER VISION

cv – topic 01

https://github.com/Waywrong/ros_course/blob/main/opencv_test

https://github.com/Waywrong/ros_course/blob/main/opencv_test/cvui.h

Product Solutions Open Source Pricing Search Sign in Sign up

Waywrong / ros_course Public

Notifications Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights

main ros_course / opencv_test / cvui.h Go to file

Waywrong cvui Latest commit f75ce2f 41 minutes ago History

1 contributor

2558 lines (2020 sloc) 101 KB Raw Blame

```
/*
A (very) simple UI lib built on top of OpenCV drawing primitives.
Version: 2.7.0

Usage:

One (and only one) of your C++ files must define CVUI_IMPLEMENTATION
before the inclusion of cvui.h to ensure its implementaiton is compiled.

E.g:

  #define CVUI_IMPLEMENTATION
  #include "cvui.h"

  int main() {
  }

All other files can include cvui.h without defining CVUI_IMPLEMENTATION.

Use of cvui revolves around calling cvui::init() to initialize the lib,
rendering cvui components to a cv::Mat (that you handle yourself) and
finally showing that cv::Mat on the screen using cvui::imshow(), which
is cvui's version of cv::imshow(). Alternatively you can use cv::imshow()
```

https://raw.githubusercontent.com/Waywrong/ros_course/main/opencv_test/cvui.h

```
/*
 A (very) simple UI lib built on top of OpenCV drawing primitives.
 Version: 2.7.0

 Usage:

 One (and only one) of your C++ files must define CVUI_IMPLEMENTATION
 before the inclusion of cvui.h to ensure its implementaiton is compiled.

 E.g:

   #define CVUI_IMPLEMENTATION
   #include "cvui.h"

   int main() {
   }

 All other files can include cvui.h without defining CVUI_IMPL

 Use of cvui revolves around calling cvui::init() to initializ
 rendering cvui components to a cv::Mat (that you handle yours
 finally showing that cv::Mat on the screen using cvui::imshow
 is cvui's version of cv::imshow(). Alternatively you can use
 to show things, but in such case you must call cvui::update()
 before calling cv::imshow().

 E.g.:

   #include <opencv2/opencv.hpp>
   #define CVUI_IMPLEMENTATION
   #include "cvui.h"
```

Save Page As...
Save Page to Pocket
Select All
Take Screenshot
View Page Source
Inspect Accessibility Properties
Inspect (Q)

```
cp   ~/Downloads/cvui.h   ~/catkin_ws/src/my_cv/src/
```

```
user@user-virtual-machine:~$ cd catkin_ws/src/my_cv/src/
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ ls
myNodeCV01.cpp  myNodeCV02.cpp
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ cp ~/Downloads/cvui.h .
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ ls
cvui.h  myNodeCV01.cpp  myNodeCV02.cpp
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ gedit myNode_ui01.cpp
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$
```

*Untitled Document 1 | myNodeCV01.cpp | myNodeCV02.cpp | myNode_ui01.cpp

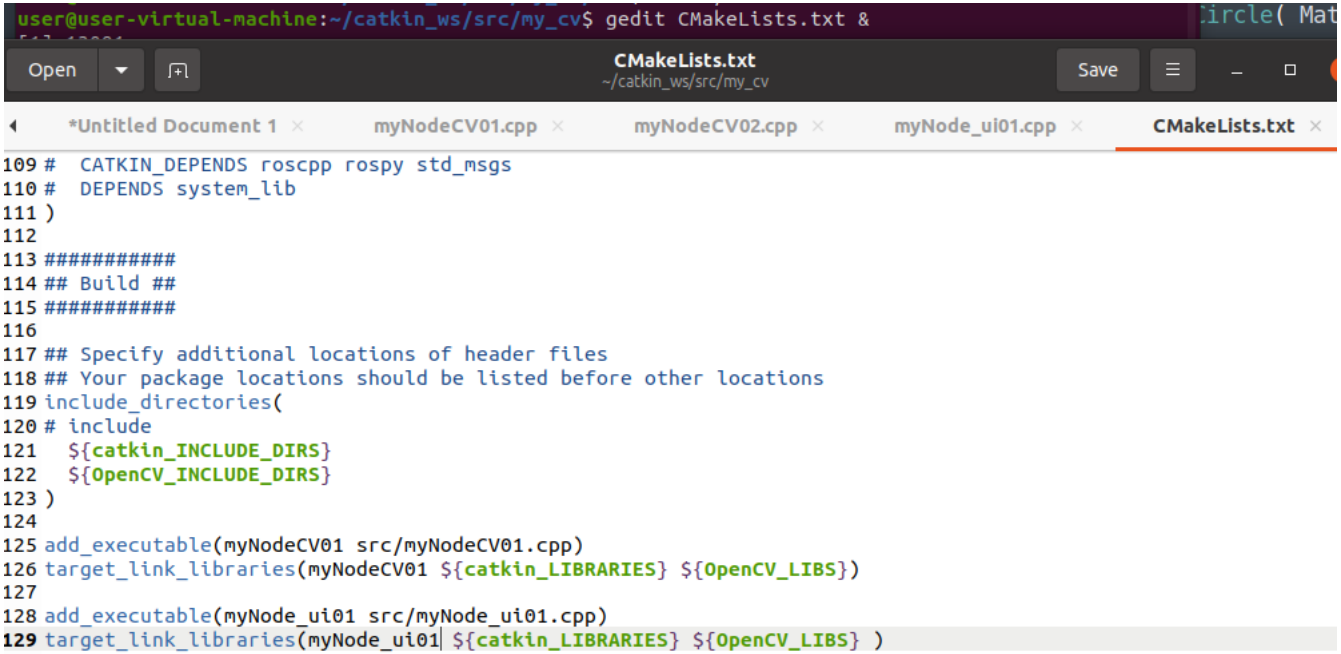
```cpp
#include "ros/ros.h"
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#define WINDOW_NAME "Button"

cv::Mat frame;
std::string btn_name = "OFF";
bool bBtn = false;
void img_Draw(void)
{
    frame = cv::Scalar(49, 52, 49);
    cvui::text(frame, 150, 50, "Button example");
    if (cvui::button(frame, 360, 80, btn_name)) {
      std::cout << "Regular button clicked!" << std::endl;
      bBtn=!bBtn;
      btn_name = bBtn? "ON":"OFF";
    }
    cvui::update();
    cv::imshow(WINDOW_NAME, frame);
    cv::waitKey(30);
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "myNode_ui01");
  ros::NodeHandle n;
  frame = cv::Mat(300, 600, CV_8UC3);
  cvui::init(WINDOW_NAME);
  ros::Rate r(30);
  while(ros::ok())
  {
    img_Draw();
    r.sleep();
    ros::spinOnce();
  }
  return 0;
}
```

```
#include "ros/ros.h"
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#define WINDOW_NAME "Button"

cv::Mat frame;
std::string btn_name = "OFF";
bool bBtn = false;
void img_Draw(void)
{
   frame = cv::Scalar(49, 52, 49);
   cvui::text(frame, 150, 50, "Button example");
   if (cvui::button(frame, 360, 80, btn_name)) {
     std::cout << "Regular button clicked!" << std::endl;
     bBtn=!bBtn;
     btn_name = bBtn? "ON":"OFF";
   }
   cvui::update();
   cv::imshow(WINDOW_NAME, frame);
   cv::waitKey(30);
}
```

```
int main(int argc, char **argv)
{
  ros::init(argc, argv, "myNode_ui01");
  ros::NodeHandle n;
  frame = cv::Mat(300, 600, CV_8UC3);
  cvui::init(WINDOW_NAME);
  ros::Rate r(30);
  while(ros::ok())
  {
    img_Draw();
    r.sleep();
    ros::spinOnce();
  }
  return 0;
}
```

find_package(OpenCV REQUIRED)

```
include_directories(
  ${catkin_INCLUDE_DIRS}
  ${OpenCV_INCLUDE_DIRS}
)
```



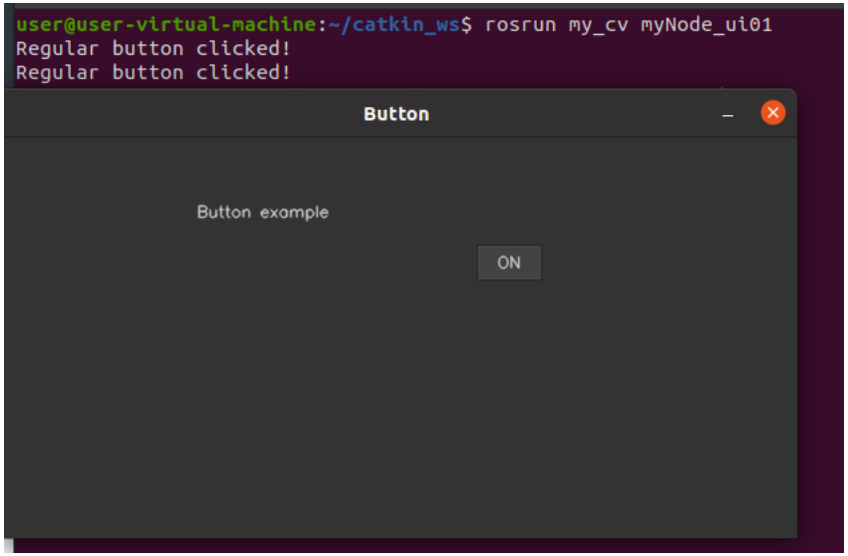

```
add_executable(myNode_ui01 src/myNode_ui01.cpp)
target_link_libraries(myNode_ui01 ${catkin_LIBRARIES} ${OpenCV_LIBS} )
```

```
cd ~/catkin_ws
catkin_make
```

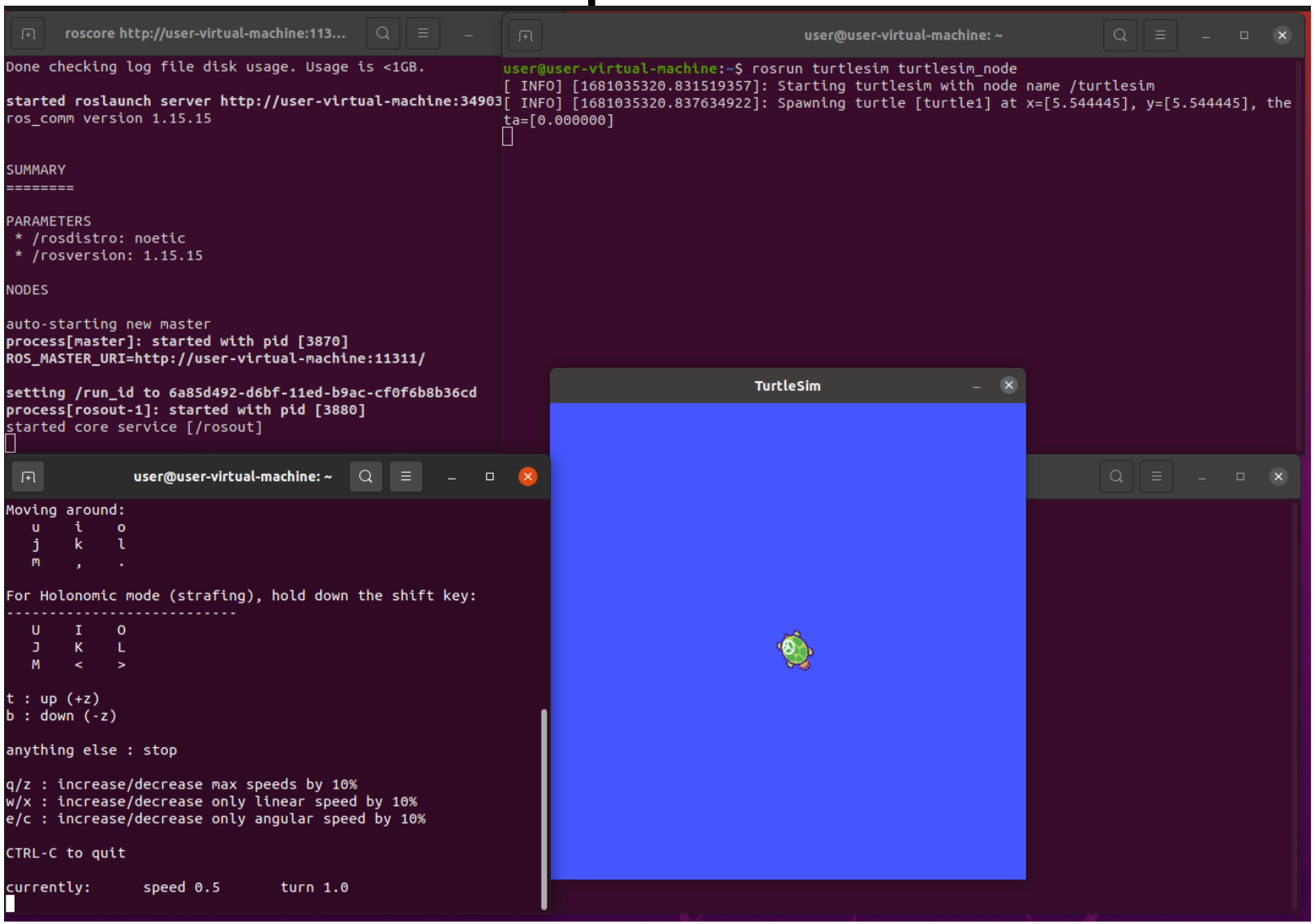
編譯程式

cv – topic 02

roscore
rosrun my_cv myNode_ui01



cv – topic 03



roscore

rosrun turtlesim turtlesim_node

rosrun teleop_twist_keyboard teleop_twist_keyboard.py /cmd_vel:=/turtle1/cmd_vel

cv – topic 04

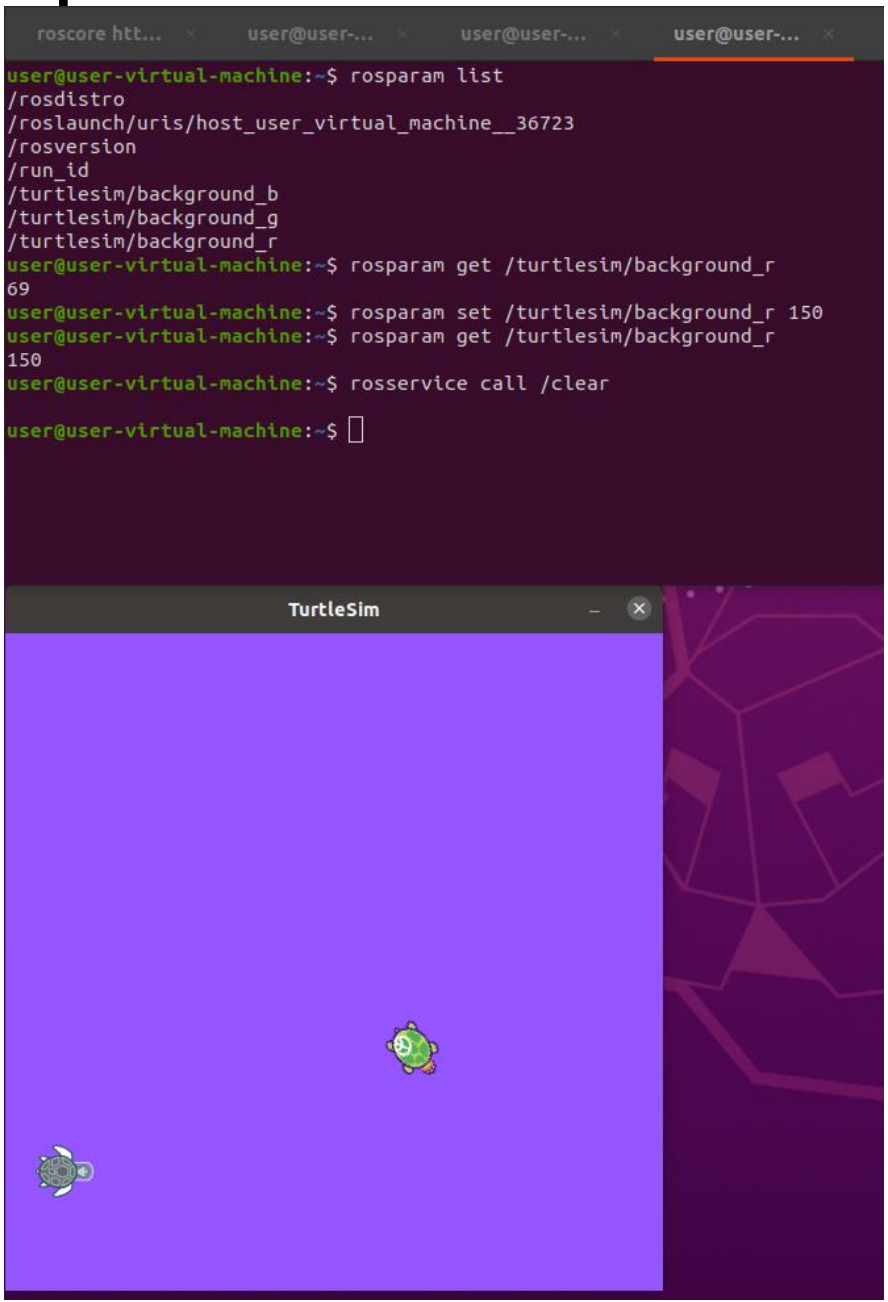
複習rosservice

rosservice list

rosservice call /spawn 1 2 0 kk

rosparam list
rosparam get /turtlesim/background_r
rosparam set /turtlesim/background_r 150
rosparam get /turtlesim/background_r
rosservice call /clear
清潔背景(強制程式讀取取ros param改變才會生效)

rosparam set /turtlesim/background_r 50
rosservice call /clear



cv – topic 05

```
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ gedit myNode_ui02.cpp &
```

Open | myNode_ui02.cpp ~/catkin_ws/src/my_cv/src | Save

myNodeCV02.cpp | myNode_ui01.cpp | CMakeLists.txt | package.xml | myNode_ui02.cpp

```cpp
#include "ros/ros.h"
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#include <geometry_msgs/Twist.h>
#define WINDOW_NAME "Button"

cv::Mat frame;
ros::Publisher cmdpub;

void my_cmd_vel(float _linear, float _angular)
{
    geometry_msgs::Twist twist;
    geometry_msgs::Vector3 linear;
    linear.x=_linear;
    linear.y=0;
    linear.z=0;
    geometry_msgs::Vector3 angular;
    angular.x=0;
    angular.y=0;
    angular.z=_angular;
    twist.linear=linear;
    twist.angular=angular;
    cmdpub.publish(twist);
}
void img_Draw(void)
{
    frame = cv::Scalar(49, 52, 49);
    cvui::text(frame, 150, 50, "Button example");
    if (cvui::button(frame, 360, 80, "up")) {
      std::cout << "up button clicked!" << std::endl;
      my_cmd_vel(1,0);
    }
    cvui::update();
    cv::imshow(WINDOW_NAME, frame);
    cv::waitKey(30);
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "myNode_ui01");
  ros::NodeHandle n;
  cmdpub = n.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel", 10);
```

```
#include "ros/ros.h"
#include <iostream>
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#define CVUI_IMPLEMENTATION
#include "cvui.h"
#include <geometry_msgs/Twist.h>
#define WINDOW_NAME "Button"

cv::Mat frame;
ros::Publisher cmdpub;

void my_cmd_vel(float _linear, float _angular)
{
   geometry_msgs::Twist twist;
   geometry_msgs::Vector3 linear;
   linear.x=_linear;
   linear.y=0;
   linear.z=0;
   geometry_msgs::Vector3 angular;
   angular.x=0;
   angular.y=0;
   angular.z=_angular;
   twist.linear=linear;
   twist.angular=angular;
   cmdpub.publish(twist);
}
```

```
void img_Draw(void)
{
   frame = cv::Scalar(49, 52, 49);
   cvui::text(frame, 150, 50, "Button example");
   if (cvui::button(frame, 360, 80, "up")) {
     std::cout << "up button clicked!" << std::endl;
     my_cmd_vel(1,0);
   }
   cvui::update();
   cv::imshow(WINDOW_NAME, frame);
   cv::waitKey(30);
}

int main(int argc, char **argv)
{
  ros::init(argc, argv, "myNode_ui01");
  ros::NodeHandle n;
  cmdpub = n.advertise<geometry_msgs::Twist>("/turtle1/cmd_vel",
10);
  frame = cv::Mat(300, 600, CV_8UC3);
  cvui::init(WINDOW_NAME);
  ros::Rate r(30);
  while(ros::ok())
  {
    img_Draw();
    r.sleep();
    ros::spinOnce();
  }
  return 0;
}
```

```
user@user-virtual-machine:~/catkin_ws/src/my_cv/src$ cd ../
user@user-virtual-machine:~/catkin_ws/src/my_cv$ gedit CMakeLists.txt &
```

Open | *CMakeLists.txt ~/catkin_ws/src/my_cv | Save

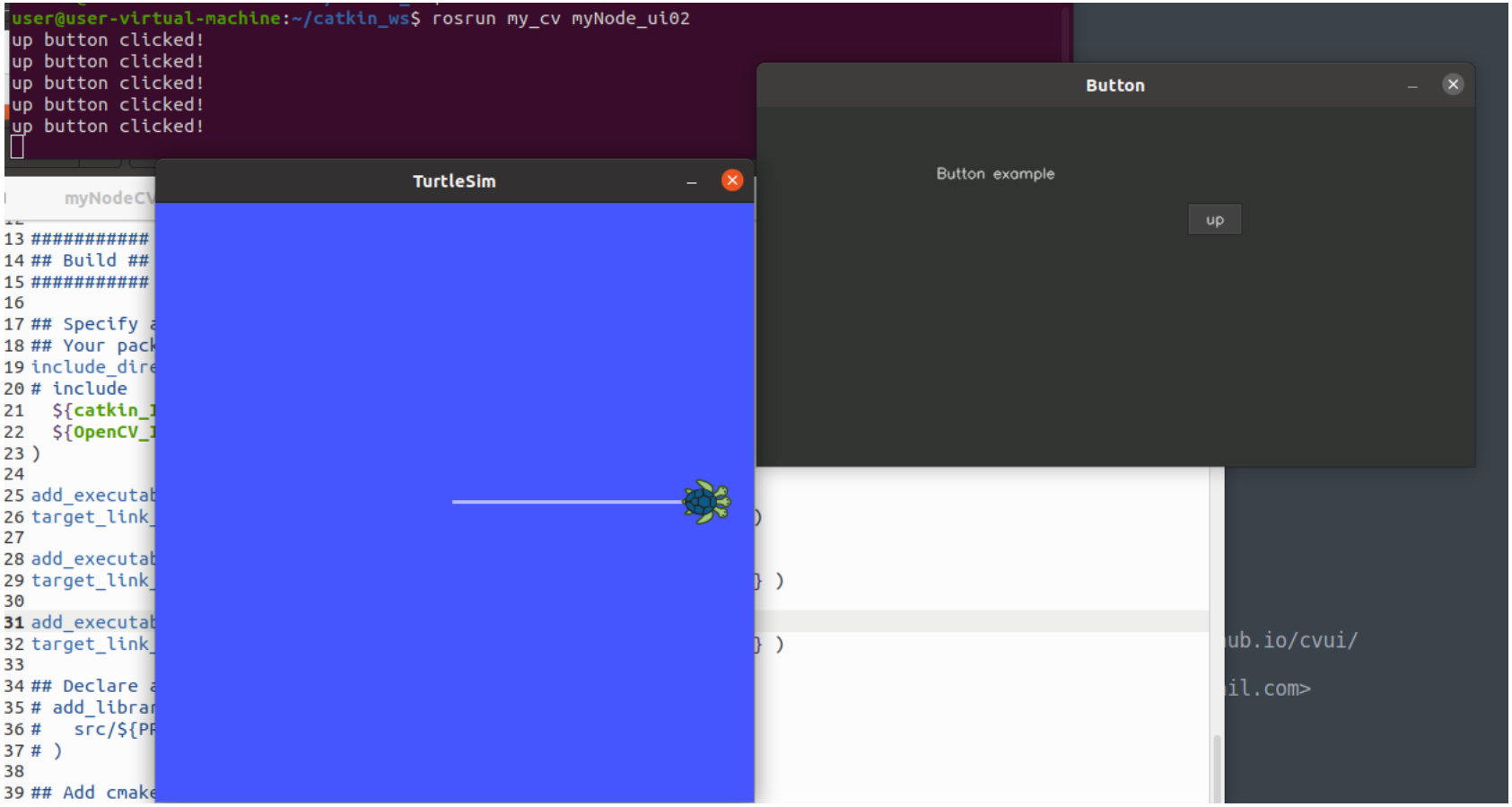
myNodeCV02.cpp | myNode_ui01.cpp | *CMakeLists.txt | package.xml | myNode_ui02.cpp

```
###########
## Build ##
###########

## Specify additional locations of header files
## Your package locations should be listed before other locations
include_directories(
# include
  ${catkin_INCLUDE_DIRS}
  ${OpenCV_INCLUDE_DIRS}
)

add_executable(myNodeCV01 src/myNodeCV01.cpp)
target_link_libraries(myNodeCV01 ${catkin_LIBRARIES} ${OpenCV_LIBS})

add_executable(myNode_ui01 src/myNode_ui01.cpp)
target_link_libraries(myNode_ui01 ${catkin_LIBRARIES} ${OpenCV_LIBS} )

add_executable(myNode_ui02 src/myNode_ui02.cpp)
target_link_libraries(myNode_ui02 ${catkin_LIBRARIES} ${OpenCV_LIBS} )
```

add_executable(myNode_ui02 src/myNode_ui02.cpp)
target_link_libraries(myNode_ui02 ${catkin_LIBRARIES} ${OpenCV_LIBS})

cd ~/catkin_ws
catkin_make 編譯程式

roscore
rosrun turtlesim turtlesim_node
rosrun my_cv myNode_ui02



練習