

Using multiple cameras

Switching the driver views

If you're interested in just switching what the driver sees, and are using SmartDashboard, the SmartDashboard CameraServer Stream Viewer has an option ("Selected Camera Path") that reads the given NT key and changes the "Camera Choice" to that value (displaying that camera). The robot code then just needs to set the NT key to the correct camera name. Assuming "Selected Camera Path" is set to "CameraSelection", the following code uses the joystick 1 trigger button state to show camera1 and camera2.

```
cs::UsbCamera camera1;
cs::UsbCamera camera2;
frc::Joystick joy1{0};
bool prevTrigger = false;
void RobotInit() override {
    camera1 = CameraServer::GetInstance()->StartAutomaticCapture(0);
    camera2 = CameraServer::GetInstance()->StartAutomaticCapture(1);
}
void TeleopPeriodic() override {
    if (joy1.GetTrigger() && !prevTrigger) {
        printf("Setting camera 2\n");
        NetworkTable::GetTable("")->PutString("CameraSelection", camera2.getName());
    } else if (!joy1.GetTrigger() && prevTrigger) {
        printf("Setting camera 1\n");
        NetworkTable::GetTable("")->PutString("CameraSelection", camera1.getName());
    }
    prevTrigger = joy1.GetTrigger();
}
```

If you're using some other dashboard, you can change the camera used by the camera server dynamically. If you open a stream viewer nominally to camera1, the robot code will change the stream contents to either camera1 or camera2 based on the joystick trigger.

```
cs::UsbCamera camera1;
cs::UsbCamera camera2;
```

Using multiple cameras

```
cs::VideoSink server;
frc::Joystick joy1{0};
bool prevTrigger = false;
void RobotInit() override {
    camera1 = CameraServer::GetInstance()->StartAutomaticCapture(0);
    camera2 = CameraServer::GetInstance()->StartAutomaticCapture(1);
    server = CameraServer::GetInstance()->GetServer();
}
void TeleopPeriodic() override {
    if (joy1.GetTrigger() && !prevTrigger) {
        printf("Setting camera 2\n");
        server.SetSource(camera2);
    } else if (!joy1.GetTrigger() && prevTrigger) {
        printf("Setting camera 1\n");
        server.SetSource(camera1);
    }
    prevTrigger = joy1.GetTrigger();
}
```

The cscore library is pretty aggressive in turning off cameras not in use. What this means is that when you switch cameras, it may disconnect from the camera not in use, so switching back will have some delay as it reconnects to the camera. This is something we would like to find a better way of doing for next year, but there is a workaround. The idea is to hook up and enable a CvSink on each camera; even if you aren't actually processing images, this fools cscore into keeping the camera connection open (as it sees multiple enabled sinks). The code for this is similar for both options above, but I'll use the SetSource() method (option 2) for illustration purposes:

```
cs::UsbCamera camera1;
cs::UsbCamera camera2;
cs::CvSink cvsink1;
cs::CvSink cvsink2;
cs::VideoSink server;
frc::Joystick joy1{0};
bool prevTrigger = false;
void RobotInit() override {
    camera1 = CameraServer::GetInstance()->StartAutomaticCapture(0);
    camera2 = CameraServer::GetInstance()->StartAutomaticCapture(1);
    server = CameraServer::GetInstance()->GetServer();
}
```

Using multiple cameras

```
// dummy sinks to keep camera connections open
cvSink1 = new cs::CvSink("cam1cv");
cvSink1.SetSource(camera1);
cvSink1.SetEnabled(true);
cvSink2 = new cs::CvSink("cam2cv");
cvSink2.SetSource(camera2);
cvSink2.SetEnabled(true);
}

void TeleopPeriodic() override {
    if (joy1.GetTrigger() && !prevTrigger) {
        printf("Setting camera 2\n");
        server.SetSource(camera2);
    } else if (!joy1.GetTrigger() && prevTrigger) {
        printf("Setting camera 1\n");
        server.SetSource(camera1);
    }
    prevTrigger = joy1.GetTrigger();
}
```

If both cameras are USB, it's worth noting that you may run into USB bandwidth limitations with higher resolutions, as in all of these cases the roboRio is going to be streaming data from both cameras to the roboRio simultaneously (for a short period in options 1 and 2, and continuously in option 3). It is theoretically possible for the library to avoid this simultaneity in the option 2 case (only), but this is not currently implemented.

Different cameras report bandwidth usage differently. The library will tell you if you're hitting the limit; you'll get this error message: "could not start streaming due to USB bandwidth limitations; try a lower resolution or a different pixel format (VIDIOC_STREAMON: No space left on device)". If you're using Option 3 it will give you this error during RobotInit(). Thus you should just try your desired resolution and adjusting as necessary until you both don't get that error and don't exceed the radio bandwidth limitations.