GRIP Computer Vision Engine

Jonathan Leitschuh (Robotics & Computer Science), Thomas Clark (Computer Science)
Advisor: Professor Brad Miller (Robotics), Professor Michael Gennert (Computer Science)

Abstract

GRIP (the Graphically Represented Image Processing engine) is an application to construct and deploy computer vision algorithms. Developing a vision program can be difficult because it is hard to visualize the intermediate results. Java and OpenCV were used to implement a graphical development tool. This simplifies and accelerates the creation of vision systems for experienced users and reduces the barrier to entry for inexperienced users. As a result, many teams with minimal computer vision knowledge successfully used our software in the 2016 FIRST Robotics Competition game.

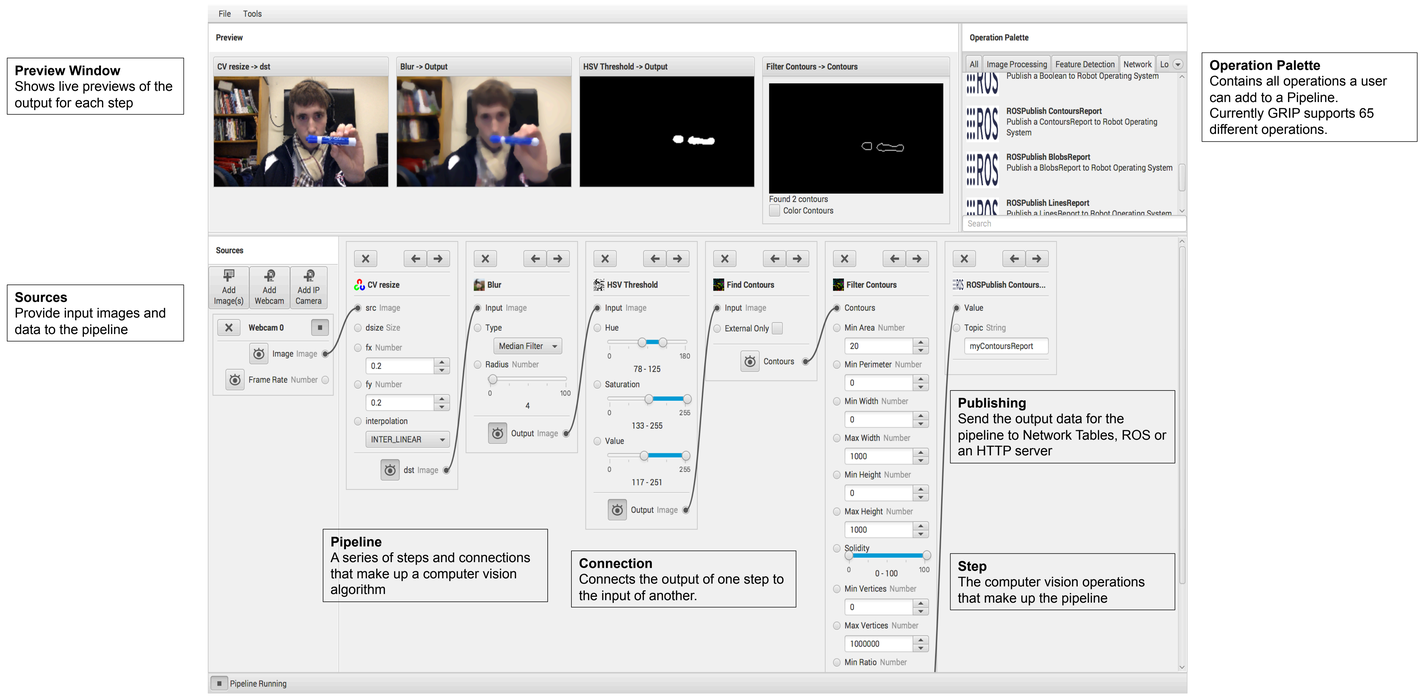
Community

GRIP is 100% Open Source under a 3 clause BSD License

- **Over 7,000 downloads on GitHub with 101 Stars & 40 Forks**
- **Many contributors from outside of WPI**
- **Very active community on Gitter**
- **Adopted by Artaic LLC. for the robotic creation of large scale mosaics**
- **FIRST reports teams using computer vision in their robots has significantly increased due to the availability of GRIP**

GRIP
computer vision engine

UI Overview



Preview Window
Shows live previews of the output for each step

Sources
Provide input images and data to the pipeline

Operation Palette
Contains all operations a user can add to a Pipeline. Currently GRIP supports 65 different operations.

Publishing
Send the output data for the pipeline to Network Tables, ROS or an HTTP server

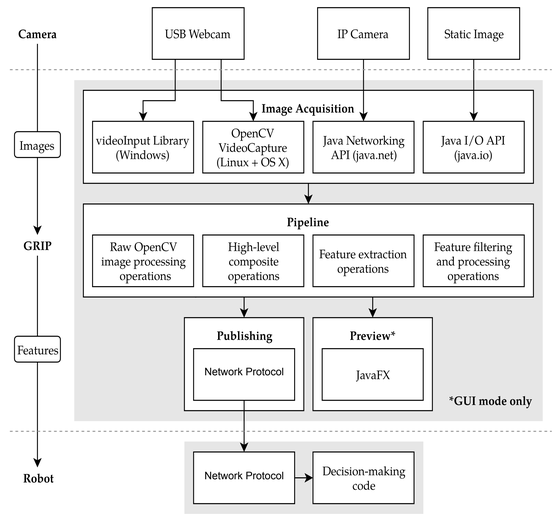
Pipeline
A series of steps and connections that make up a computer vision algorithm

Connection
Connects the output of one step to the input of another.

Step
The computer vision operations that make up the pipeline

Architecture

GRIP is divided into two modules - the core and the graphical user interface (GUI). The core is responsible for performing the actual computer vision. The GUI allows the user to both manipulate the set of operations and preview the outputs of the core.



Build System

- Automated build/release system with Gradle
- All pull requests tested and approved by Travis CI & AppVeyor
- Static code analysis by Codacy
- Compile time error checking with Error Prone
- Automated Junit tests for the Core Module
- Automated UI Tests with TestFX
- Code coverage metrics with Jacoco and CodeCov
- Automatic releases published by Travis CI & AppVeyor
- Full integration with Gitter chat
- Dependency version monitoring with VersionEye

Supported Operating Systems

- **Windows (x86/64)**
- **Mac OSX**
- **Linux (Ubuntu & Fedora)**
- **Embedded Linux ARM (Headless)**