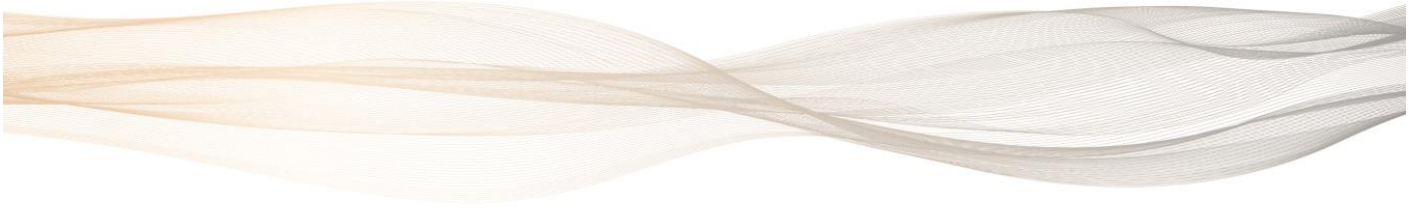


VividFlow



User Manual

Group 9



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Autumn – Spring 2016

Project Details

Subject	CSIT321 – Project
Session	Autumn – Spring 2016
Subject Coordinator	Mark Freeman
Campus	Wollongong

Group Members

Student #	Name	Email
4773810	Joshua Coleman	jic224@uowmail.edu.au
4752405	Philip Edwards	pme446@uowmail.edu.au
4669174	Thomas Nixon	tn941@uowmail.edu.au

Client

Dr Igor Kharitonenko

Faculty of Engineering and Information Sciences

School of Computing & Information Technology

Table of Contents

INTRODUCTION	3
1 GENERAL: SITE ELEMENTS	4
1.1 THE CONTEXT-BASED NAVIGATION MENU	4
2 USERS.....	5
2.1 CREATING AN ACCOUNT	5
2.2 LOGGING IN.....	6
2.3 LOGGING OUT.....	6
2.4 ACCOUNT MANAGEMENT	7
2.4.1 <i>Resetting password</i>	7
2.4.2 <i>Disabling account</i>	7
3 RESOURCES.....	8
3.1 UPLOADING A RESOURCE	8
3.2 RENAMING A RESOURCE	9
3.3 DELETING A RESOURCE	10
3.4 REPLACING A RESOURCE	10
3.5 DOWNLOADING A RESOURCE.....	11
4 MODULES	12
4.1 INTRODUCTION TO THE MODULE DESIGNER.....	12
4.2 CREATING A MODULE	13
4.2.1 <i>Adding and removing sockets</i>	13
4.2.2 <i>Generating module code</i>	14
4.2.3 <i>Using the module header/library to write code</i>	15
4.2.4 <i>Publishing and saving a module</i>	16
4.3 MODIFYING AN EXISTING MODULE	17
4.4 DELETING AN EXISTING MODULE	17

4.5	ACCESSING THE MODULE LIBRARY FOR OFFLINE DEVELOPMENT	17
5	ALGORITHMS.....	18
5.1	ALGORITHM LIST	18
5.1.1	Overview.....	18
5.1.2	Open	18
5.1.3	Create	18
5.1.4	Delete.....	18
5.2	ALGORITHM DESIGNER	19
5.2.1	Overview.....	19
5.2.2	Algorithm Name	20
5.2.3	Module Selector.....	20
5.2.4	Algorithm Canvas	21
5.2.6	Inserting Nodes.....	24
5.2.7	Editing Nodes.....	26
5.2.8	Deleting Nodes	26
5.2.9	Linking Nodes.....	27
5.2.10	Save an Algorithm	28
5.2.11	Run an Algorithm	28
5.2.12	View Algorithm Output	28

Introduction

VividFlow is a web-application with the purpose of providing an interface for rapid prototyping of computer vision algorithms by allowing researchers to compose discrete modules of self-contained functionality that can be reused in multiple algorithms.

This document is a manual for researchers which are unfamiliar with the system and require insight as to its use. The manual envelopes all expected interaction intended for a researcher, and is divided into five (5) main sections. Anything from creating an account to prototyping an algorithm from scratch is covered. The table of contents on the previous page provides an overview of the content available, and is helpful for indexing the location to specific questions a researcher may have.

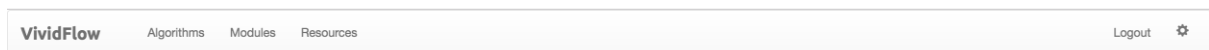
1 General: site elements

1.1 The context-based navigation menu

The context-based navigation menu located at the top of the application is the primary interface to navigation to specific functionality. It is context-based in that the buttons available from the navigation menu change depending on your current location.

The system allows multiple researchers to concurrently access the system and work on their algorithms. Researchers can upload C++ programs they have written with the assistance of a helper library that is provided with the system. A researcher can define what inputs and outputs a module will produce. Once they have defined a module, it can be placed in the algorithm graph using the algorithm designer. The system can then run the algorithm and will allow the researcher to download the outputs that were produced by the algorithm.

The following image depicts top-level navigation that exists as a base for all subsequent contexts:



Only the two top-level pages Algorithms and Modules contain additional contexts that extend from the aforementioned base. The additional elements in each contexts are as outlined in the following images:

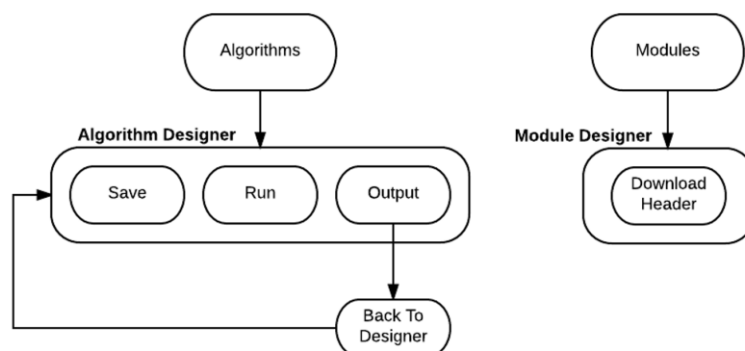
Algorithm



Module



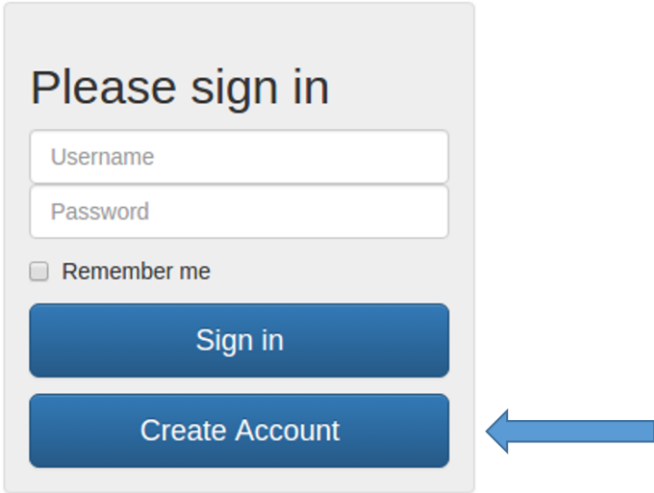
Navigation of the additional contexts are depicted in the following diagram:



2 Users

2.1 Creating an account

Creating an account is as simple as completing the form at the site's index (see image below). The form consists of two input fields that accepts a username and password, respectively. On completion, engage the "Create Account" button to create a new user under the specified credentials.



Please sign in

Username

Password

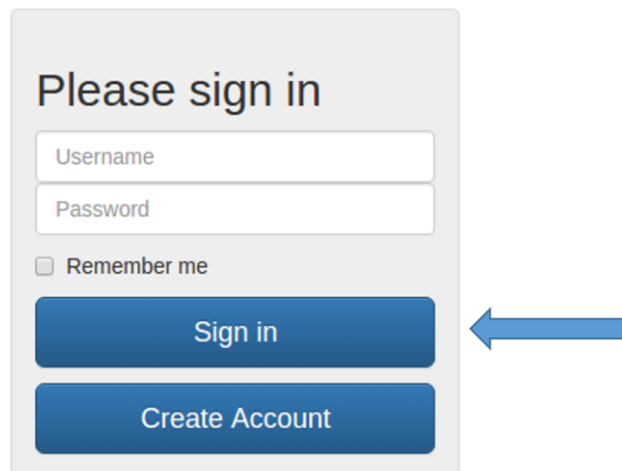
☐ Remember me

Sign in

Create Account

2.2 Logging in

The simplicity of logging in is equivalent to that of creating an account (see section 2.1). At the site's index, you are presented a form that consists of two input fields that accept a username and password, respectively. Complete the form, entering in the credentials that were entered upon creating an account, and engage the "Sign in" button.



Please sign in

Username

Password

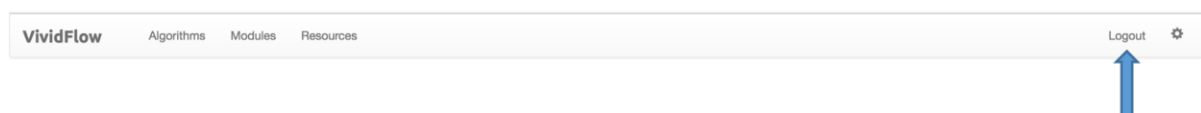
☐ Remember me

Sign in

Create Account

2.3 Logging out

Once logged in the user is redirected to a new index that contains a **context-based navigation menu**, as shown in the image below. To the rightmost section of the navigation menu, there exists a "Logout" button that, when engaged, terminates a user's session and redirects him/her to the index that contains the login form.

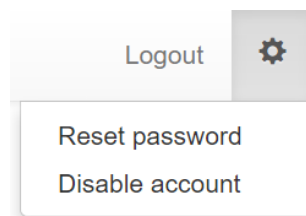


2.4 Account management

Account-related functionality concerned within this section includes:

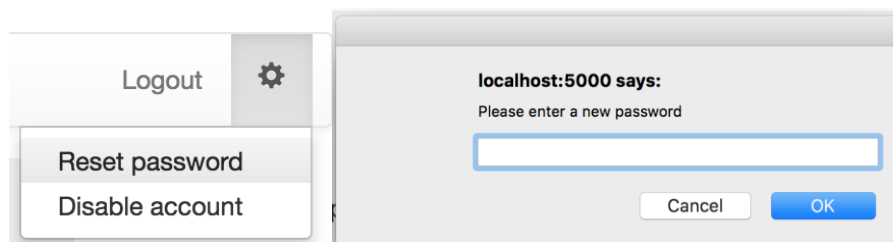
- Resetting password
- Disabling account

Firstly, the user must be logged in. To access any of the aforementioned functionality, simply navigate to and toggle the cogwheel icon located at rightmost section of the navigation bar. A drop-down menu will appear. For specific information regarding the options contained in the drop-down menu, refer to sections 2.4.1 and 2.4.2 for resetting passwords and disabling accounts, respectively.



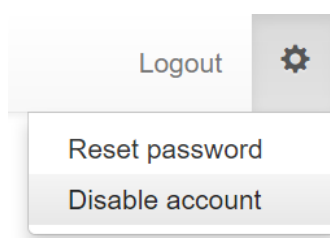
2.4.1 Resetting password

Select the "Reset password" listing from the drop-down menu. The user will then be prompted with a dialog that contains a form that contains an input field and two buttons, "OK" and "Cancel". In the input field, enter the new password desired and hit "OK". The user's password is subsequently updated.



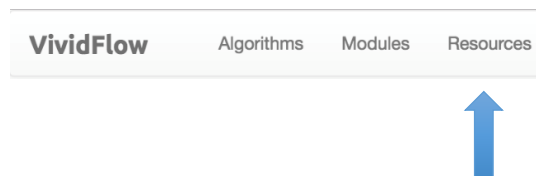
2.4.2 Disabling account

Select the "Disable account" listing from the drop-down menu. Upon pressing, the user's session will be terminated, and the user will be redirected to the site's index pre-login.



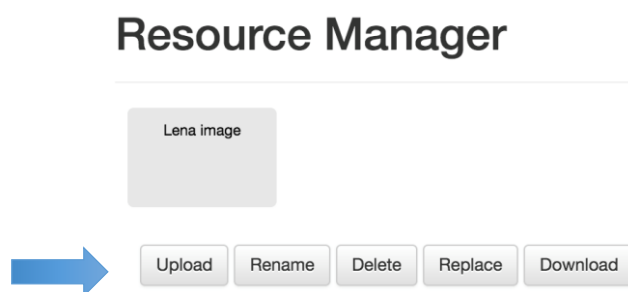
3 Resources

The resource manager can be accessed by clicking “Resources” from the context-based navigation menu. In the resource manager files such as videos, images and other data files can be uploaded and given a name. These resources can be placed in algorithms.

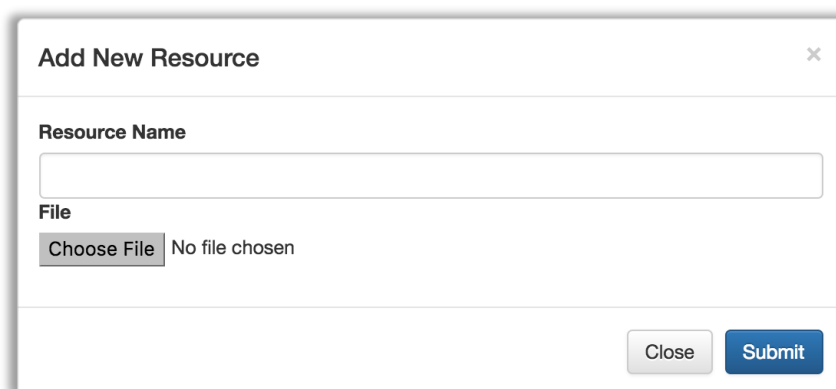


3.1 Uploading a resource

By clicking the “Upload” button the system will present a form where a new file can be selected and a friendly name can be given to the resource. Resources do not need a type specified as files can be interpreted by the modules as required.



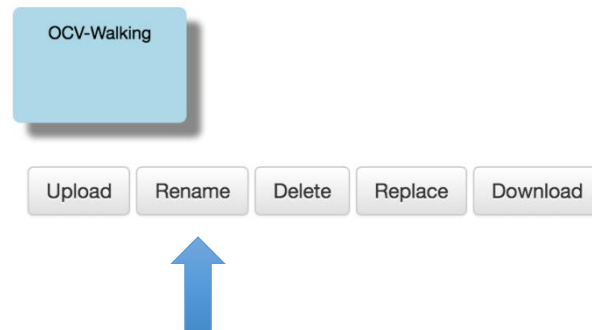
To upload the file and commit the resource to the system click “Submit”. Please be patient when uploading large files. When the resource has finished uploading the page will refresh and the new resource will be visible. Alternatively, clicking “Close” will hide the pop up form and not commit any changes.

A modal dialog box titled 'Add New Resource' with a close button (X) in the top right corner. It contains a 'Resource Name' label followed by a text input field. Below that is a 'File' label and a 'Choose File' button. To the right of the 'Choose File' button, it says 'No file chosen'. At the bottom right of the dialog, there are two buttons: 'Close' and 'Submit'.

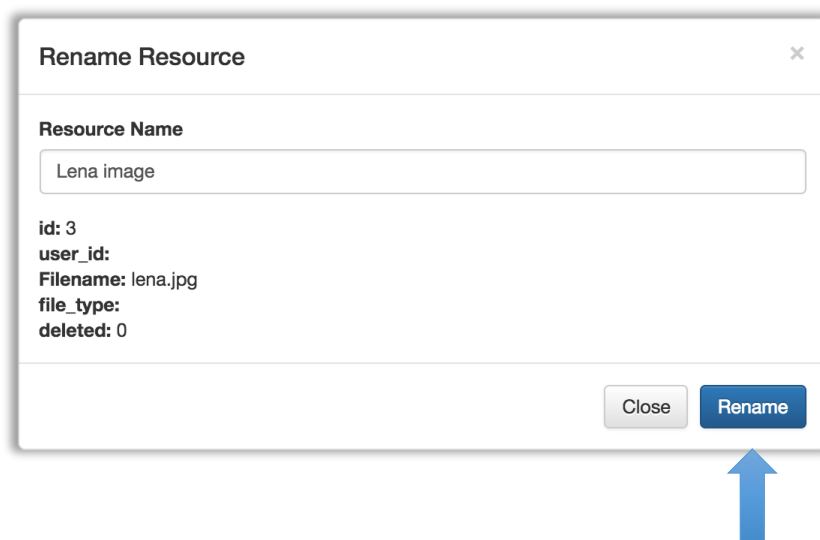
3.2 Renaming a resource

Click the resource that is to be renamed and it will turn blue to indicate that it is selected. Click “Rename” and the system will display a form where a new resource name can be entered.

Resource Manager



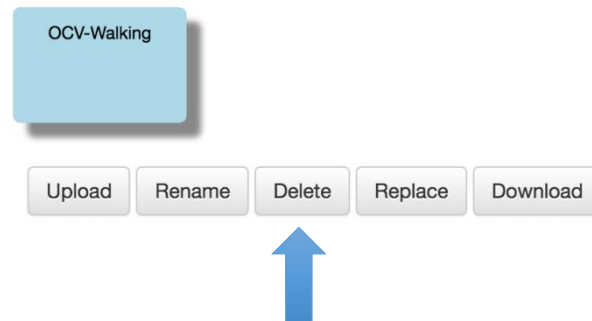
Update the text box with the new name and then click “Rename”. Alternatively, clicking “Close” will hide the pop up form and not commit any changes.



3.3 Deleting a resource

Click the resource that is to be deleted and it will turn blue to indicate that it is selected. Click “Delete” and the item will be deleted. Please be careful as there is no confirmation.

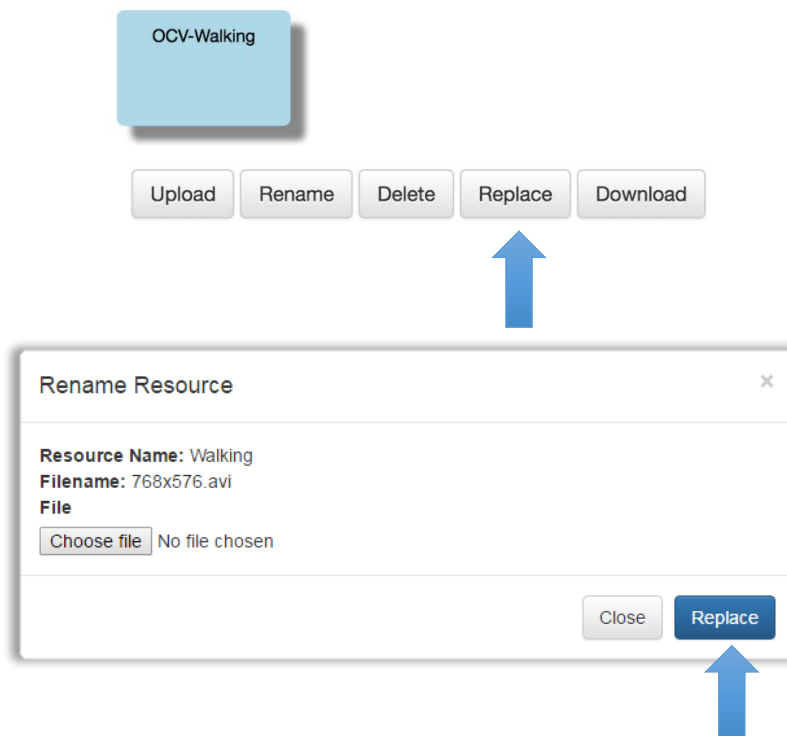
Resource Manager



3.4 Replacing a resource

Replacing a resource allows the content of an existing resource to be updated while keeping all references in the system updated. Click the resource that is to be updated and it will turn blue to indicate that it is selected. Alternatively, clicking “Close” will hide the pop up form and not commit any changes.

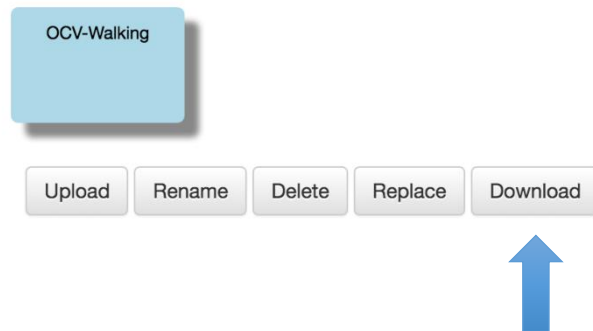
Resource Manager



3.5 Downloading a resource

An existing resource can be downloaded in its original form by selecting the resource and then clicking “Download”.

Resource Manager



4 Modules

4.1 Introduction to the module designer

Module Designer

Module Name:

UntitledNode

Module Code:

```
1 #include <vividflow.h>
2
3 int main(int argc, char *argv[])
4 {
5     initInterface(argc, argv);
6
7     // Socket code will be generated here. Do not delete.
8
9     freeInterface();
10
11     return 0;
12 }
13
```

Input Sockets:

Arg#	Name	Type
------	------	------

Add Socket

Output Sockets:

Arg#	Name	Type
------	------	------

Add Socket

Generate Code

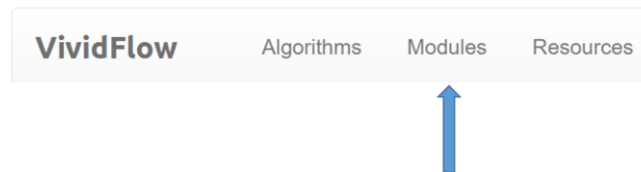
Publish Module

Save Module

The module designer is the interface in which all properties relative to a module is defined, which include the module's name, code, and input and output sockets, respectively. A placeholder for the module's name is provided, and a code template is presented to get you started. For more information regarding the template, see code generation at section 4.2.2.

4.2 Creating a module

Firstly, navigate to the module list by engaging the **Modules** button located in the **context-based navigation menu**.



From the module list, the user may create a new module by engaging the “Create Module” button. Upon pressing, the user will be presented with the module designer introduced in section 4.1, the interface in which modules are defined.

Module List



4.2.1 Adding and removing sockets

The interface for defining a socket includes a table structure with three elements:

1. An uneditable argument number that increments on each addition and is relative to type (input or output).
2. An input field for the socket’s name.
3. A drop-down menu of possible data types a socket may accept.

From the same interface, sockets may also be removed.

Input Sockets:

Arg#	Name	Type	
1	<input type="text" value="inSocket1"/>	<div>AnyDataType ▾</div>	✕
2	<input type="text" value="inSocket2"/>	<div>String ▾</div>	✕

Output Sockets:

Arg#	Name	Type	
1	<input type="text" value="outSocket1"/>	<div>Integer ▾</div>	✕

4.2.2 Generating module code

Upon defining the input and/or output sockets required for the module, one may optionally generate code that will provide an interface (setup variables) that relate to each defined socket, which are separated distinctly with the use of comments and newlines. Furthermore, an overview of the socket definitions given will be displayed in comments at the top of your code.

Module Code:

```
1 // Input sockets:
2 // 1. inSocket1 is of type AnyDataType.
3 // 2. inSocket2 is of type String.
4
5 // Output sockets:
6 // 1. outSocket1 is of type Integer.
7
8 #include <vividflow.h>
9
10 int main(int argc, char *argv[])
11 {
12     initInterface(argc, argv);
13
14     // Input sockets:
15     struct GenericData *inSocket1 = readGeneric(1);
16     char *inSocket2 = readString(2);
17
18     // Output sockets:
19     writeInteger(1, /* Output data here. */);
20
21     freeInterface();
22
23     return 0;
24 }
25
```


4.2.3 Using the module header/library to write code

The following table presents the primary functionality offered by the VividFlow module library for researchers to interface. These functions provide the basis for interfacing input/output sockets with ease. For a stronger overview, please refer to the source code, which is accessible as described in section 4.5.

Function	Purpose
<i>ModuleInterface</i> <i>*initInterface</i> (<i>int</i> argc, <i>char</i> *argv[])	Initialise the interface: setup sockets for use. This should only be called within once and before all subsequent functionality from the module library.
<i>void freeInterface(void)</i>	Release resources: deallocate memory/objects used to represent and manage sockets. This should only be called once and after all former references to functionality from the module library.
<i>GenericData *</i> <i>readGeneric(int</i> argNum)	Read from input socket <i>argNum</i> of type <i>AnyDataType</i> , <i>Image</i> , <i>Video</i> , or <i>Text</i> .
<i>char *</i> <i>readString(int</i> argNum)	Read from input socket <i>argNum</i> of type <i>String</i> .
<i>int</i> <i>readInteger(int</i> argNum)	Read from input socket <i>argNum</i> of type <i>Integer</i> .
<i>double</i> <i>readFloat(int</i> argNum)	Read from input socket <i>argNum</i> of type <i>Float</i> .
<i>void writeGeneric</i> (<i>int</i> argNum, <i>const</i> <i>GenericData *data</i>)	Write parameter <i>data</i> to output socket <i>argNum</i> of type <i>AnyDataType</i> , <i>Image</i> , <i>Video</i> , or <i>Text</i> .
<i>void writeString</i> (<i>int</i> argNum, <i>const char</i> <i>*string</i>)	Write parameter <i>string</i> to output socket <i>argNum</i> of type <i>String</i> .
<i>void writeInteger</i> (<i>int</i> argNum, <i>const int</i> <i>number</i>)	Write parameter <i>number</i> to output socket <i>argNum</i> of type <i>Integer</i> .
<i>void writeFloat</i> (<i>int</i> argNum, <i>const double</i> <i>number</i>)	Write parameter <i>number</i> to output socket <i>argNum</i> of type <i>Float</i> .

<i>char</i> <i>*getInputSocketFilename</i> <i>(int argument)</i>	Acquire a pointer to the char buffer that contains the filename of the input socket given by the parameter <i>argument</i> , which refers to the socket's argument number.
<i>char</i> <i>*getOutputSocketFilename</i> <i>(int argument)</i>	Acquire a pointer to the char buffer that contains the filename of the output socket given by the parameter <i>argument</i> , which refers to the socket's argument number.
<i>GenericData *readGeneric</i> <i>(int argNum)</i>	Read from input socket <i>argNum</i> of type <i>AnyDataType</i> , <i>Image</i> , <i>Video</i> , or <i>Text</i> .
<i>char *readString(int argNum)</i>	Read from input socket <i>argNum</i> of type <i>String</i> .
<i>int readInteger(int argNum)</i>	Read from input socket <i>argNum</i> of type <i>Integer</i> .
<i>double readFloat(int argNum)</i>	Read from input socket <i>argNum</i> of type <i>Float</i> .

4.2.4 Publishing and saving a module

Located at the absolute bottom of the module designer there are two buttons, "Publish Module" and "Save Module". Saving a module by engaging the "Save Module" button will cause it to preserve the status of **In development**, i.e. a user should save when their module is incomplete and he/she will need to come back at a later time to proceed with development. Publishing a module can be achieved by engaging the "Publish Module" button, which will update the status of the module to **Published**. A published module is considered complete. Any subsequent publishing on published module will cause it to be **Retired**, which effectively deletes the original module and creates an entirely new one.

Publish Module

Save Module

4.3 Modifying an existing module

Modifying an existing module can be achieved by clicking on the cell with the module name of choice to modify from the module list, which will redirect the developer to the module designer with the latest properties defined. These properties are subject to change of one's own volition. Upon modification of any properties of the module, publish or save the module (see section 4.2.4) to commit your changes; otherwise, any changes will not be preserved.

Module List



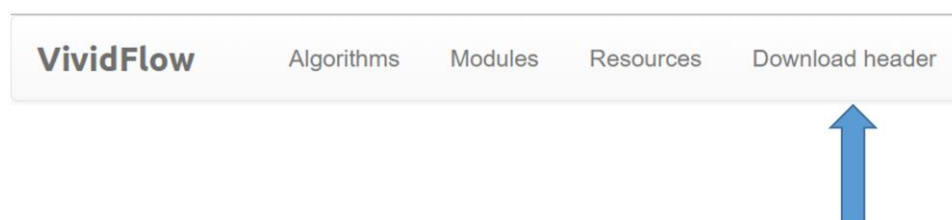
4.4 Deleting an existing module

Removing an existing, unwanted module is as simple as visiting the module list and engaging the cross (X) button dedicated to removal.



4.5 Accessing the module library for offline development

Developing offline is as simple as navigating to the module designer, regardless of the module, and engaging the rightmost element of the site's **context-based navigation menu** dedicated to presenting the header file containing the code.

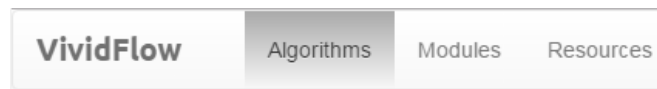


5 Algorithms

5.1 Algorithm List

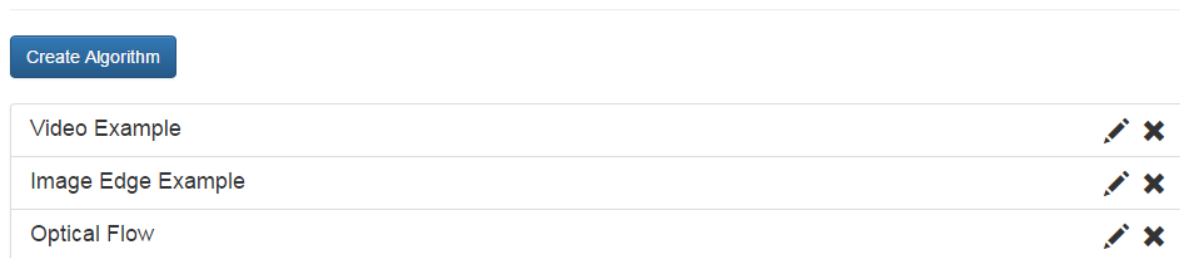
5.1.1 Overview

The Algorithm List displays all the algorithms that a user has created. From the Algorithm List a user can create, open and delete algorithms. The Algorithm List can be accessed from the site's **context-based navigation menu** by clicking on "Algorithms".



The list shows each algorithm name.

Algorithm List



5.1.2 Open

Each algorithm shown can be clicked on, this will open it in the Algorithm Designer. From there the algorithm can be modified, run and output viewed.

5.1.3 Create

A new algorithm can be created by clicking on the "Create Algorithm" button. Once clicked the user will be taken to the Algorithm Designer. The new algorithm will be blank and ready to be developed. It is best to change the name of the algorithm from the default "UntitledAlgorithm".

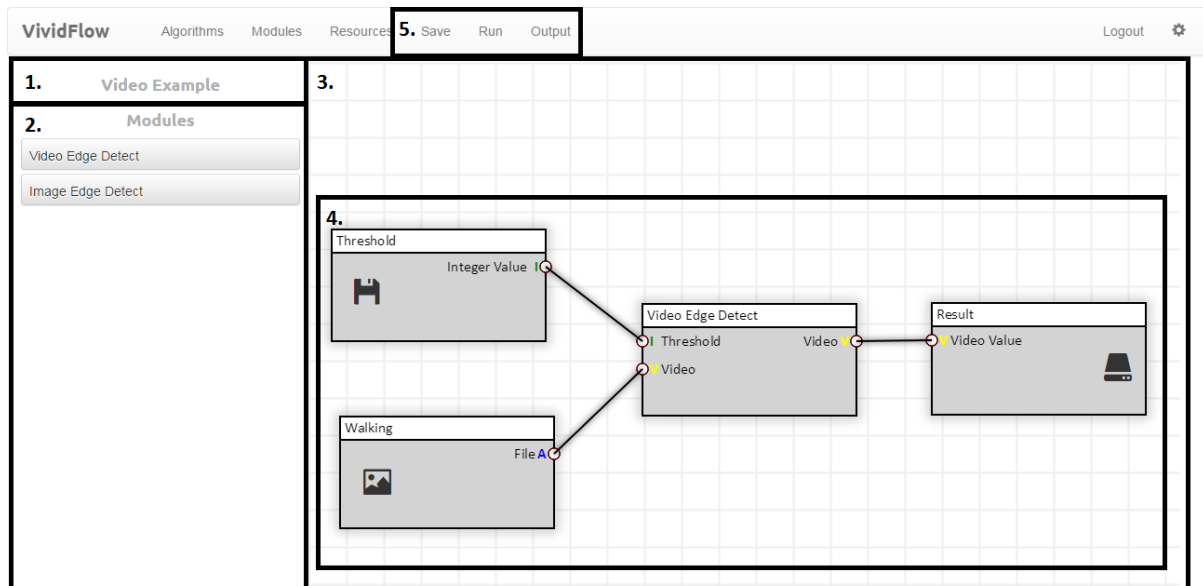
5.1.4 Delete

The cross next to the algorithm allows a user to delete an algorithm. When clicked the user will be prompted to confirm the deletion. Once deleted the user cannot access any of the output from the algorithm. The modules and resources will still be able to be used in a new algorithm (since these are stored separately).

5.2 Algorithm Designer

5.2.1 Overview

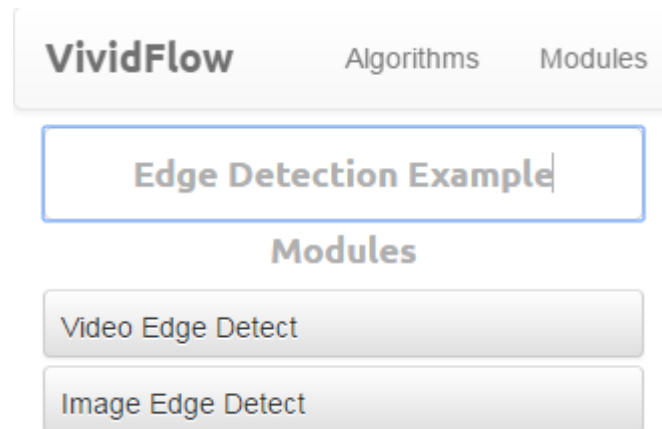
The Algorithm Designer allows a user to modify, run and view output from an algorithm. It gives the user access to all the components to achieve this. These different components are shown in the following screenshot of the Algorithm Designer.



1. Algorithm Name
2. Module Selector
3. Algorithm Canvas
4. Algorithm (nodes and links)
5. Algorithm Functions

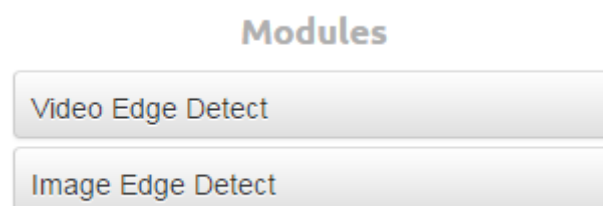
5.2.2 Algorithm Name

The name of the Algorithm can be changed by clicking on the current name and editing the text. This is the name that will be shown in the Algorithm List so it is best to make it something unique. This should be the first thing changed in an algorithm so that it can easily be identified from the algorithm list.



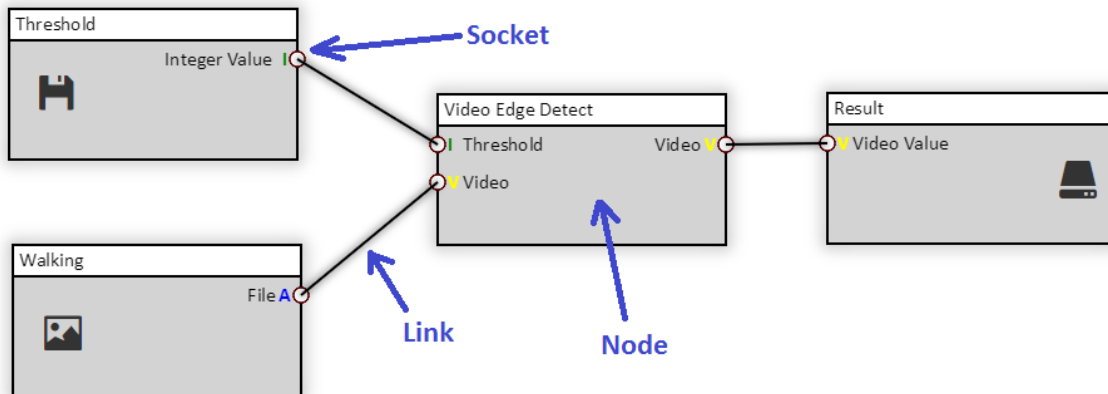
5.2.3 Module Selector

The module selector lets a user insert a Module Node into the algorithm. When a module button is clicked on the Module Node will be added to the centre of the Algorithm Canvas. The list of modules shown are those that the user has already created. To add another module to the list a user needs to create a new module in the Module List.



5.2.4 Algorithm Canvas

The Algorithm Canvas is where an algorithms design is shown and modified. The nodes and links that make up the algorithm can be modified as well as inserting new nodes and links.



Dragging Canvas

The algorithm can be moved around by left clicking (and holding) on the background grid and moving the mouse. When finished moving, release the mouse button. The position of the algorithm will reset when the algorithm is opened again.

Dragging Nodes

Nodes can be moved around the algorithm by left clicking (and holding) on the node and then moving the mouse. The node will become green while it is selected. To finish moving release the mouse button. A user can only move one node at a time. Links will automatically resize to where the new node is.

Context Menus

To bring up the context menu the user can right click anywhere on the algorithm canvas. The menu will change depending on what is under the mouse at the time. To close a context menu the user can click somewhere else in the algorithm canvas. The following elements will show these options:

- **Value Node:** Edit, Delete Node
- **Output Node:** Edit, Delete Node
- **Resource Node:** Delete Node
- **Module Node:** Delete Node
- **Link:** Delete Link
- **Background:** Add Value, Add Resource, Add Output

Node

A node can either be a value, resource, output or module. Nodes have input and/or outputs. These are called sockets, nodes are connected with sockets.

Socket

Sockets have different types and these will depend on what the input or output is required. Sockets are only compatible with the same socket type (except for AnyDataType) and invalid links will be restricted by this type. Each type of socket has a colour coded letter to make identifying socket types easier. The different types that a socket can be are:

- **U**: UnsupportedDataType
- **A**: AnyDataType
- **S**: String
- **I**: Integer
- **F**: Float
- **I**: Image
- **V**: Video
- **T**: Text

Link

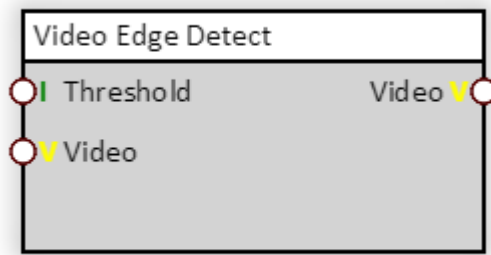
A node can be joined to another by connecting two sockets. The connection is shown by a line to each socket, this line is a Link. The link allows output from one node to become the input for the next. For example, a resource node of an image can be linked to a module node that takes an image as input and converts it to black and white image. A link can only connect compatible sockets.

5.2.5 Types of Nodes

A node is one of the core building blocks of an algorithm. All nodes have at least one socket. A socket allows for input and/or output from a node. A node can be one of four types, and each type has a specific use.

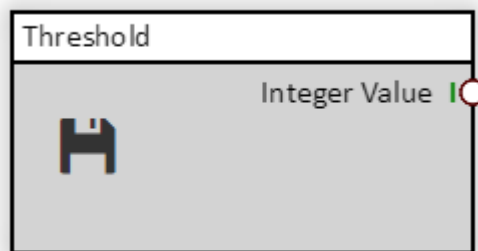
Module Node

A module node has the properties of a module. It has input and output sockets that were defined in the Module Designer. To have the output of a module saved the output socket/s need to be linked into an output node.



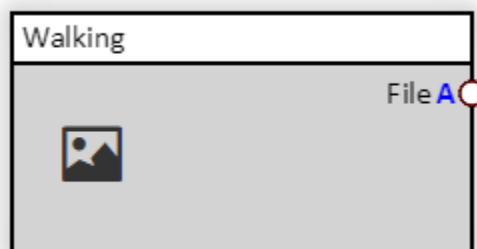
Value Node

A value node holds a value of a certain type that can be easily changed. A value is useful to feed values into a module. A value can be an integer, float, string or an AnyDataType. The value node only has a single output socket.



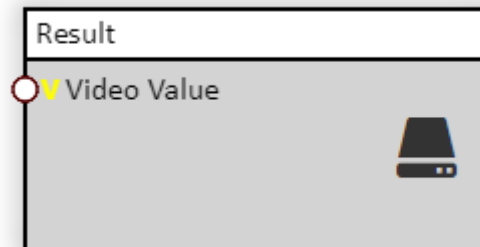
Resource Node

The resources uploaded in the resource manager can be added to an algorithm as a resource node. These can be used to feed a file into a module node. When creating a resource node, the user will be able to select a resource from the resource library. The resource node only has a single output socket.



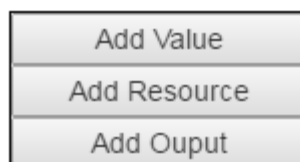
Output Node

Output nodes are used to view output from an algorithm. Each output node will generate data that can be viewed after an algorithm has run. The type of the value node can be specified. If a user wants to see the output of a module, then an output node can be linked to a modules output socket. An output node only has one input socket.



5.2.6 Inserting Nodes

A module node can be inserted by clicking on a module from the Module Selector. The value, resource and output nodes can be inserted by right clicking on the algorithm canvas grid and selecting the one you want. You will be able to configure the node and then a new node will be inserted where the mouse was clicked.



Module Node

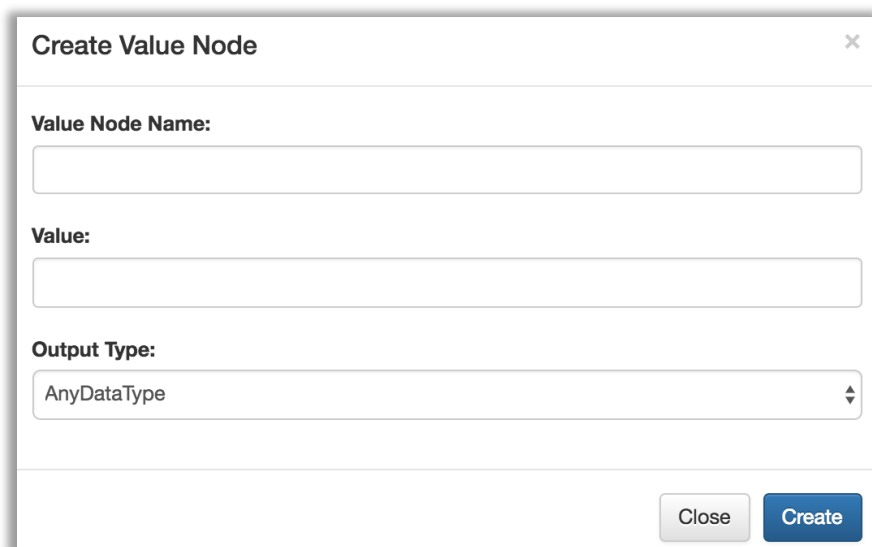
Clicking on a module from the Module Selector will add a new module node to the centre of the algorithm canvas. The node can then be moved and linked. There are no configuration options for the module in the Algorithm Designer. Any changes that the user wants to apply must be done in the Module Designer.

Value Node

When creating a new value node, the user will be prompted for:

- Value Node Name: the name shown at the top of a node
- Value: the value that the node will output
- Output Type: the type of the socket that will be used for output

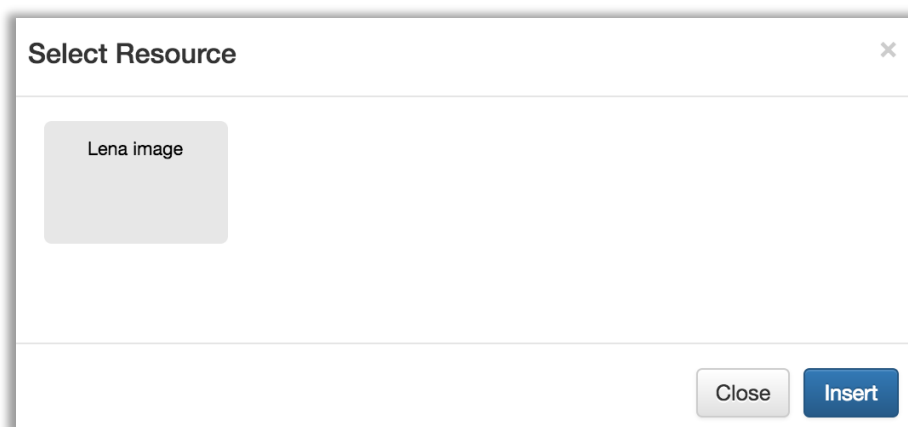
Once the fields have been completed the user can click the “Create” button to insert the new Value Node into the algorithm. If the user decides not to insert the node, the “Close” button can be clicked.



The "Create Value Node" dialog box features a title bar with a close button (X). It contains three input fields: "Value Node Name:" (a text box), "Value:" (a text box), and "Output Type:" (a dropdown menu currently showing "AnyDataType"). At the bottom right, there are two buttons: "Close" and "Create".

Resource Node

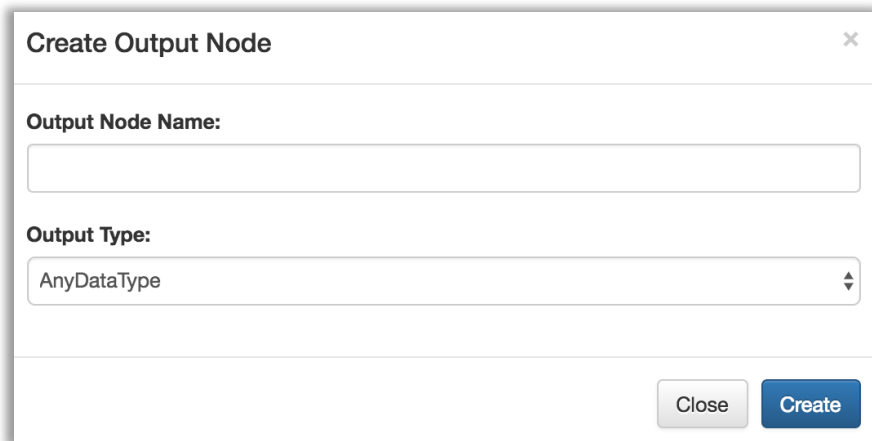
When inserting a resource node, the user will be prompted with a selection of resources that they have uploaded. If the user wants to add more resources this can be done in the Resource Manager. The desired resource can be clicked on to select it. The “Insert” button will add the resource to the algorithm. The name of the node will become the name of the resource. If the user decides not to insert the node, the “Close” button can be clicked.



The "Select Resource" dialog box has a title bar with a close button (X). It displays a single resource as a gray box labeled "Lena image". At the bottom right, there are two buttons: "Close" and "Insert".

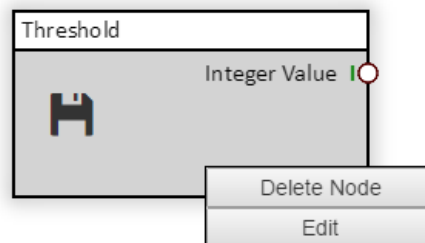
Output Node

When inserting an output node, the user needs to give the output node a name and select which type of data it will receive. Clicking the “Create” button will add the output node to the algorithm. If the user decides not to insert the node, the “Close” button can be clicked.

A dialog box titled "Create Output Node" with a close button (X) in the top right corner. It contains two main sections: "Output Node Name:" with a text input field, and "Output Type:" with a dropdown menu currently showing "AnyDataType". At the bottom right, there are two buttons: "Close" and "Create".

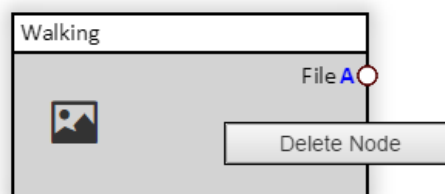
5.2.7 Editing Nodes

The output node and the value node can be modified. This can be done by right clicking on the node and selecting “Edit”. The user will be prompted with the same form that they used when creating the node. Any of the fields can be changed and then updated.



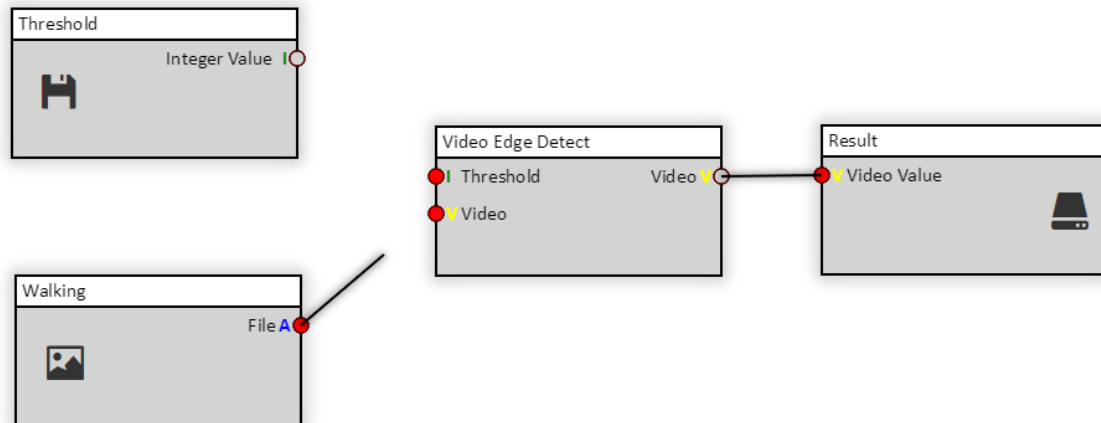
5.2.8 Deleting Nodes

A node can be deleted from the algorithm by right clicking on the node and selecting “Delete Node”. When the node is deleted it will also delete the links attached to its sockets.

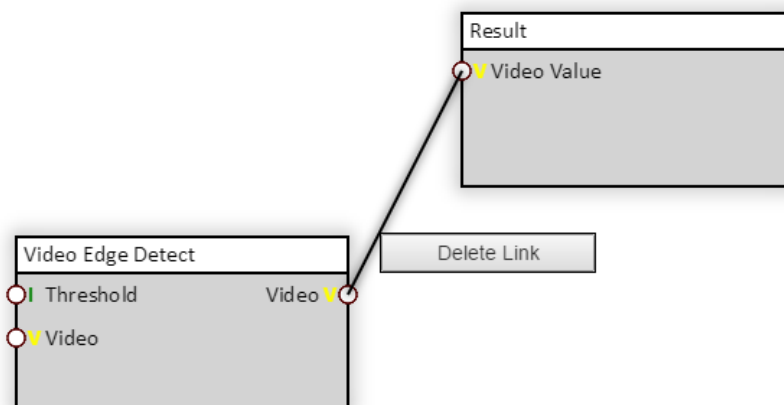


5.2.9 Linking Nodes

To create a link between two sockets, the user needs to left click and drag from one socket to another. Sockets that are compatible with the one the user starts from will be highlighted in red, making selection easier. Sockets may not be compatible if they are of different types (excluding AnyDataType) or if they do not make an input to output link (it cannot have an output socket to another output socket).



Links can be deleted by right clicking on the link and selecting delete. Links will also be deleted if one of the nodes they are attached to is deleted.



5.2.10 Save an Algorithm

To save an algorithm the “Save” button in the **context-based navigation menu** can be clicked. A prompt will confirm that the algorithm has been saved. If the user does not save any changes made will be lost. The user should save after any modification to the algorithm before:

- Navigating away from the page
- Running the algorithm
- Viewing the output of the algorithm

5.2.11 Run an Algorithm

To schedule the algorithm to run the user can click the “Run” button in the **context-based navigation menu**. A prompt will confirm that the algorithm has been scheduled to run. To view the status or output of the run the user can click on “Output”. The algorithm will run the last saved version of the algorithm.

5.2.12 View Algorithm Output

When an algorithm has been scheduled to run, a task is created on the Algorithm Output page. There are three different states that a task can be in:

- **Pending:** The task is waiting to start, it may be waiting for other tasks to finish first.
- **In Progress:** The task is running now
- **Complete:** The task has finished running and output can be viewed

The tasks are also colour coded to make it easier to identify where they are up to. The meaning of the colours are

- **Yellow:** Task is “Pending” or “In Progress”
- **Green:** Task is “Complete”

The columns of the task table can be sorted to make it easy to find a task. To sort a column the user can click on the column heading. Clicking on the column heading again will sort it in the opposite order.

Algorithm Output

Show 10 entries Search:

State	Start	Finish	Return Code
Completed	2016-10-09 16:03:07	2016-10-09 16:03:16	0
Completed	2016-10-13 13:40:31	2016-10-13 13:40:39	0
InProgress	2016-10-13 13:43:51	None	0
Pending	None	None	0

Showing 1 to 4 of 4 entries Previous 1 Next

When a task is complete the user can click on the row. This will bring up a modal displaying the log and output from the completed algorithm run. The output can be clicked on to view/download. The output shown here is determined by the output nodes in the algorithm. If another output is required to be viewed, then create a new output node in the algorithm.

