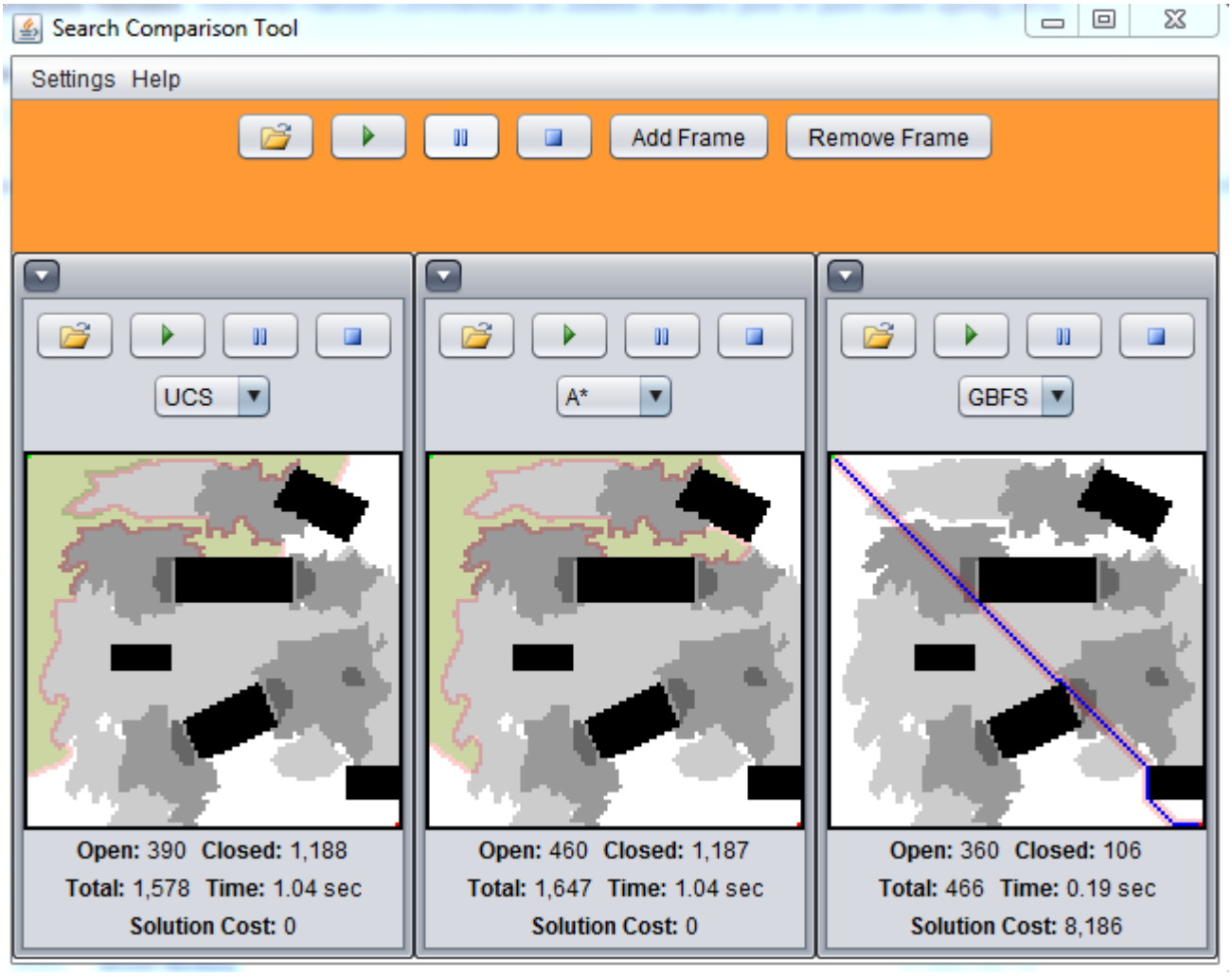
Search Algorithm App

User guide

Search Algorithm App



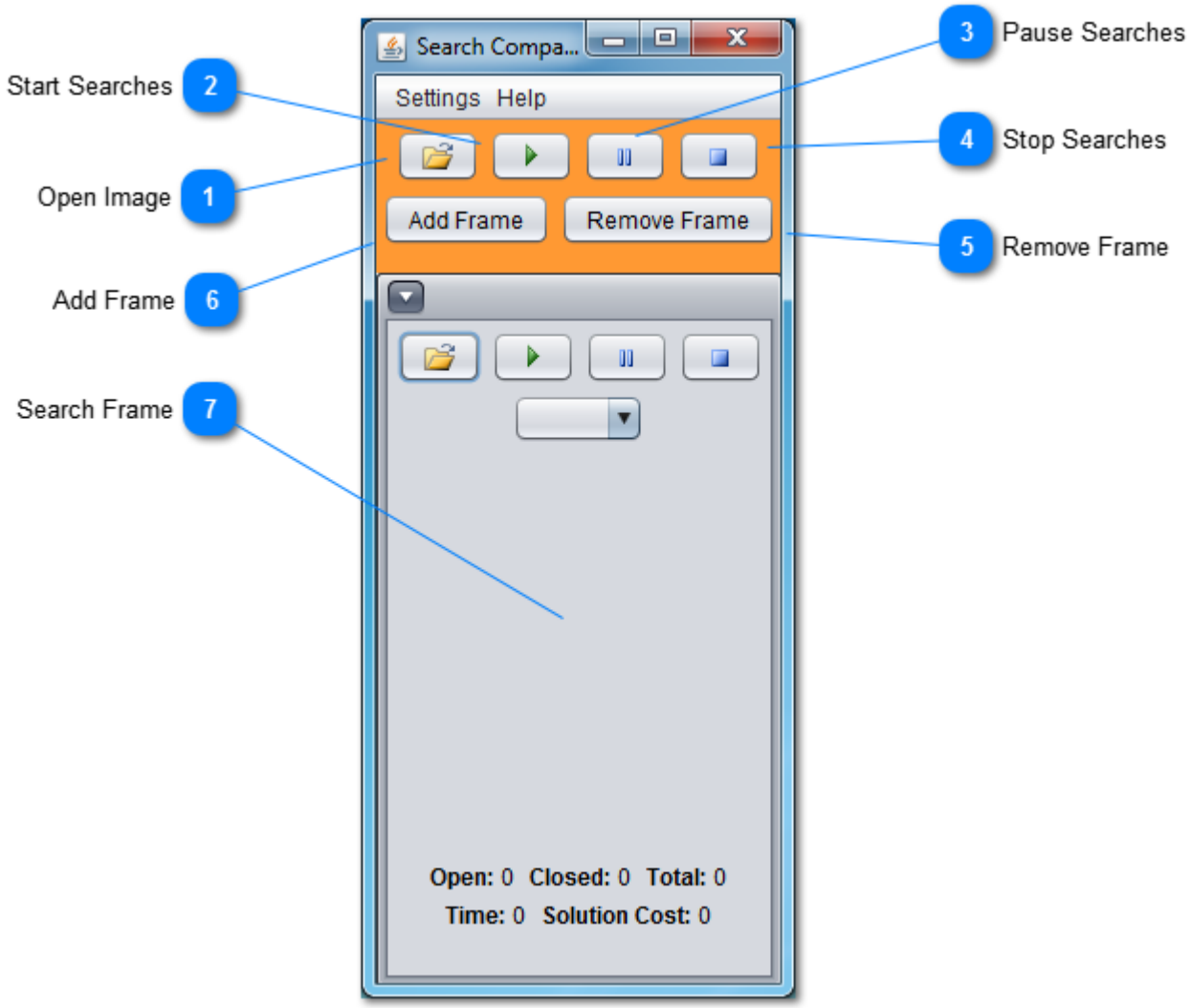
A tool designed to illustrate the differences between various search algorithms.

Created by Chad Oftedahl oftesoft@gmail.com

Control Help

Any questions about what any of the controls do can be found here

Main Window



1 Open Image

Loads an image into all Seach Frames. The images that work with the program are any image that is black and white and has a solid black border of at least 1 pixel thickness.

2 Start Searches

Starts the searches in all Search Frames that have an image loaded and a search algorithm selected. The seach begins at the selected start point (green pixel) and will run until all the nodes are searched or the end point (red pixel) is found.

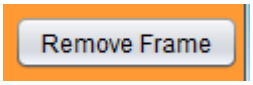
3 Pause Searches

Starts the searches in all Search Frames that have an image loaded and a search algorithm selected. The seach begins at the selected start point (green pixel)

4 Stop Searches

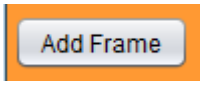
Stops all searches that are running. Any search that is restarted after being stopped will start over from the beginning.

5 Remove Frame



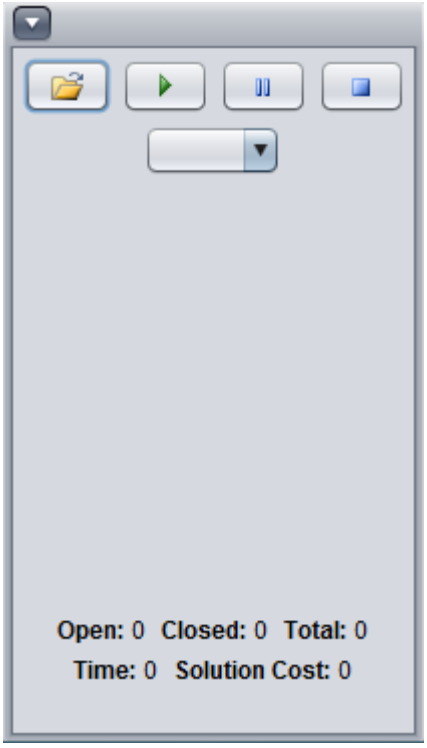
Removes the most recently added search frame and resizes the remaining frames to fill up as much of the main window as possible

6 Add Frame



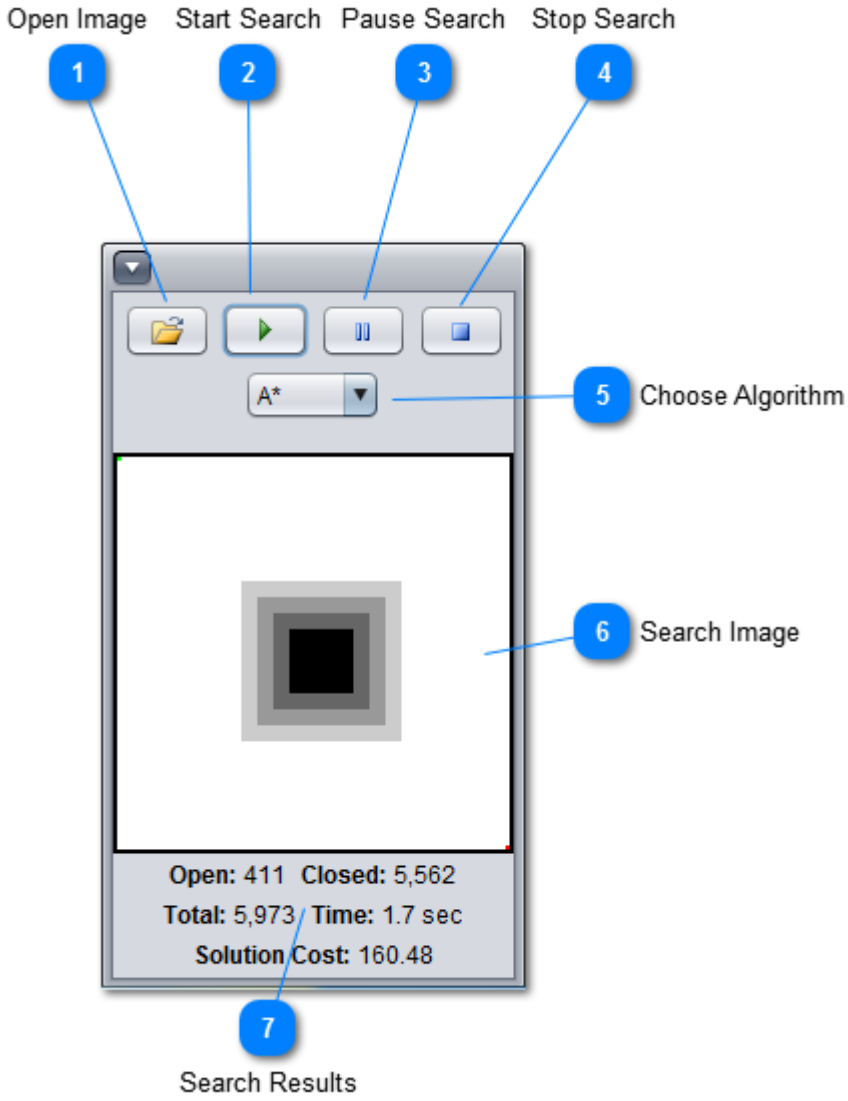
Adds a frame to the main window with the same image / starting points as the most recently added frame. If neccesary all of the Search Frames will be resized to fit into the main window. Any Search Frame without an image loaded will not be part of the resizing process.

7 Search Frame



Container for one image that will be searched by the selected search algorithm. Information about the controls in the Search Frame can be found here.

Search Frame



1 Open Image

Loads an image into the Seach Frame. The images that work with the program are any image that is black and white and has a solid black border of at least 1 pixel thickness.

2 Start Search

Starts the currently selected search on the currently loaded image. The seach begins at the selected start point (green pixel)

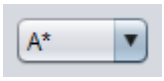
3 Pause Search

Pauses any search currnetly in progress. If a search is started after it is paused then the search picks up from where it was paused at.

4 Stop Search

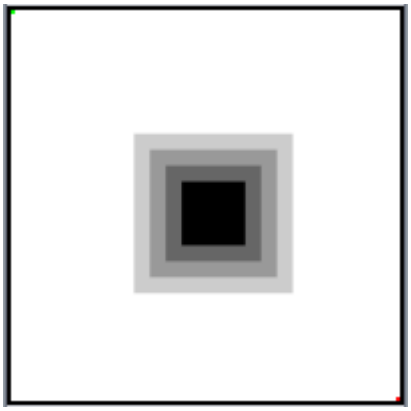
 Stops any search currently in progress. If a search is started after it is stopped the search starts from the beginning.

5 Choose Algorithm



Selects the search algorithm to be used

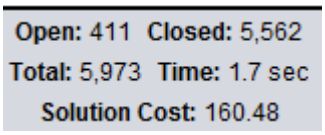
6 Search Image



The image that the search algorithm will run on. The start point of the search will be the green pixel and the goal point of the search will be the red pixel. By default the start point is placed in the top left corner of the image and the goal point is placed in the bottom right.

To change the starting point quickly click on any non black pixel in the image.
To change the ending point click on any non black pixel in the image AND hold down the mouse button for about two seconds before releasing it.

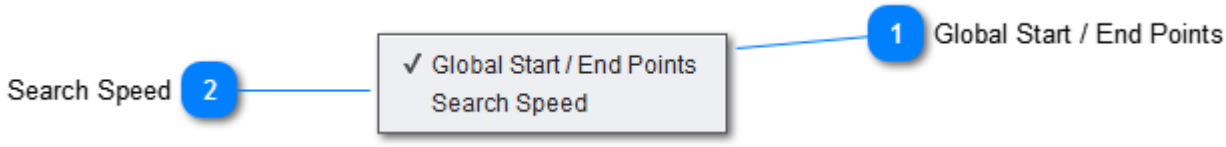
7 Search Results



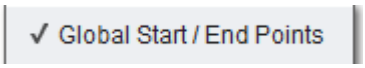
Once a search finishes (finds the end point) or is interrupted by the user the current statistics of the search are displayed in this area. The meanings are as follows

- Open - The number of nodes currently in the open list
- Closed - The number of nodes currently in the closed list
- Total - The nuber of nodes in the open and closed lists
- Time - The time in seconds since the search was started. This will probably differ from the actual run time due to operating system thread scheduling.
- Solution Cost - The cost of the path found from the starting point to the ending point. The cost of individual nodes (i.e. pixels) is calculated accoring to the formula (Cost = 256 - pixel grayscale value). On the extremes white pixels will have a cost of 1 and black pixels will have a cost of 256. Black pixels are considered impassible and will therefore never be part of a solution path. The images used by the program must have a border of black pixels to "contain" the search within the image.

Settings Menu

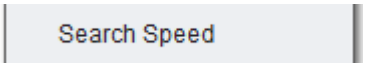


1 Global Start / End Points



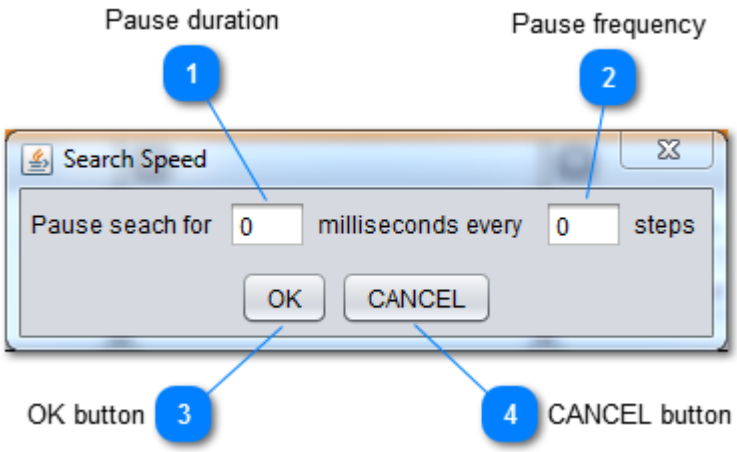
If this menu item is checked then when a start or end point is changed in any search frame the changes will take place in all search frames. This is useful when comparing a bunch of different algorithms on the same image. If this is NOT checked then changes to any search frames start or end points will not affect any other search frame.

2 Search Speed

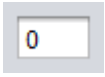


Opens up the Search Speed dialog which allows the speed of the search to be changed.

Search Speed dialog

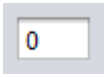


1 Pause duration



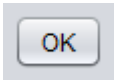
The length of time (in milliseconds) that the search will pause

2 Pause frequency



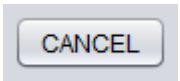
The number of search steps run before the search will pause.

3 OK button



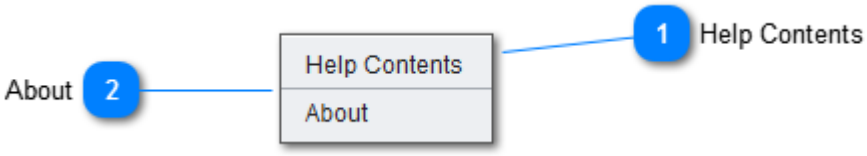
Saves the entered Pause duration and Pause frequency values and closes the dialog window

4 CANCEL button

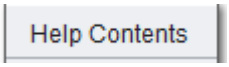


Discards any changes and closes the dialog window

Help Menu

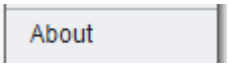


1 Help Contents



Shows this PDF help file

2 About



Shows information about the program and the author

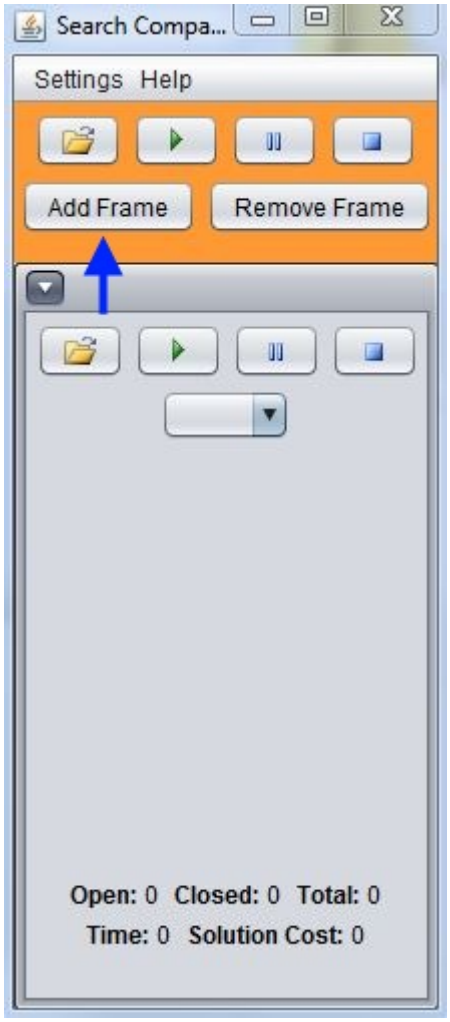
Quick Start Guide

The following steps will get you started in getting a search up and running as well as teach you the basics of using the program

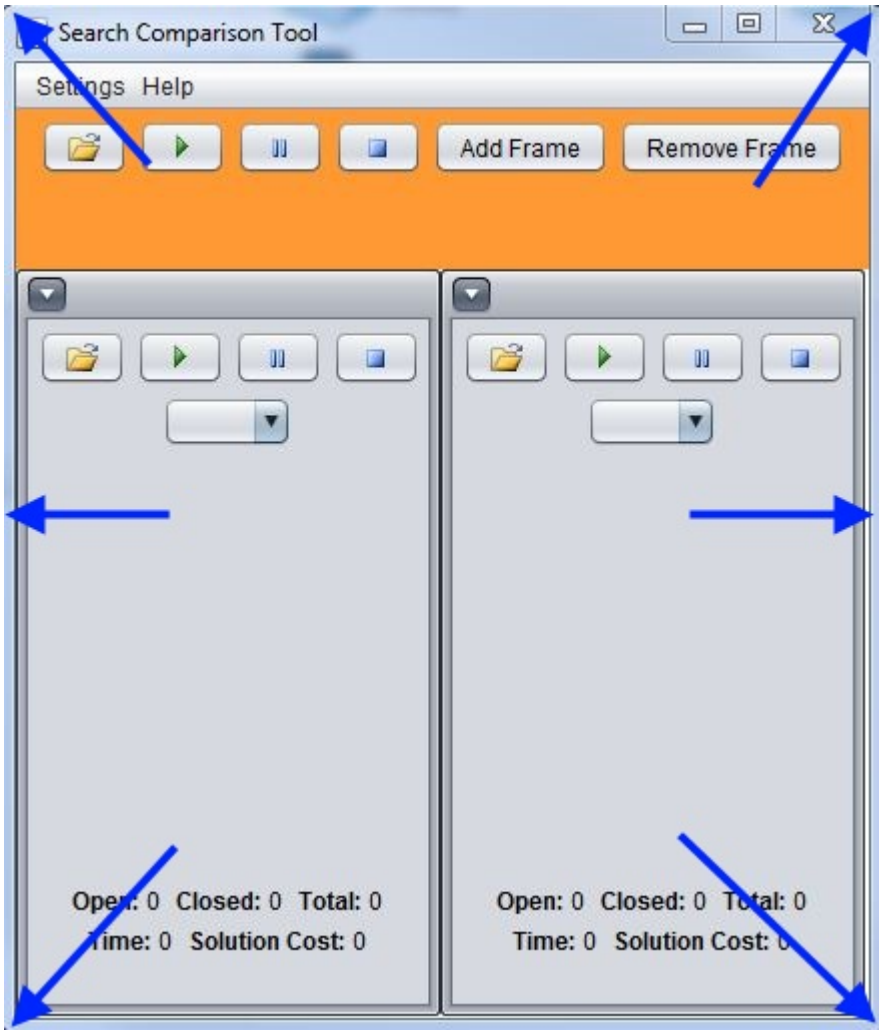
1. Adding a Search Frame
2. Loading an image
3. Setting start / end points
4. Selecting a search algorithm
5. Starting a search

Adding a Search Frame

The reason this application is called the Search Comparison App is that it's meant to run more than one search at a time so you can compare algorithms. For each algorithm you want to compare you will need a Search Frame in which to run the search. To add more Search Frames simply click the "Add Frame" button to add another Search Frame to the main window.



If you do not see another search frame as shown below, you will have to increase the size of the main window by clicking on one of the edges pointed to by the blue arrows and dragging it to increase the main window size.



The next step is to load an image to search

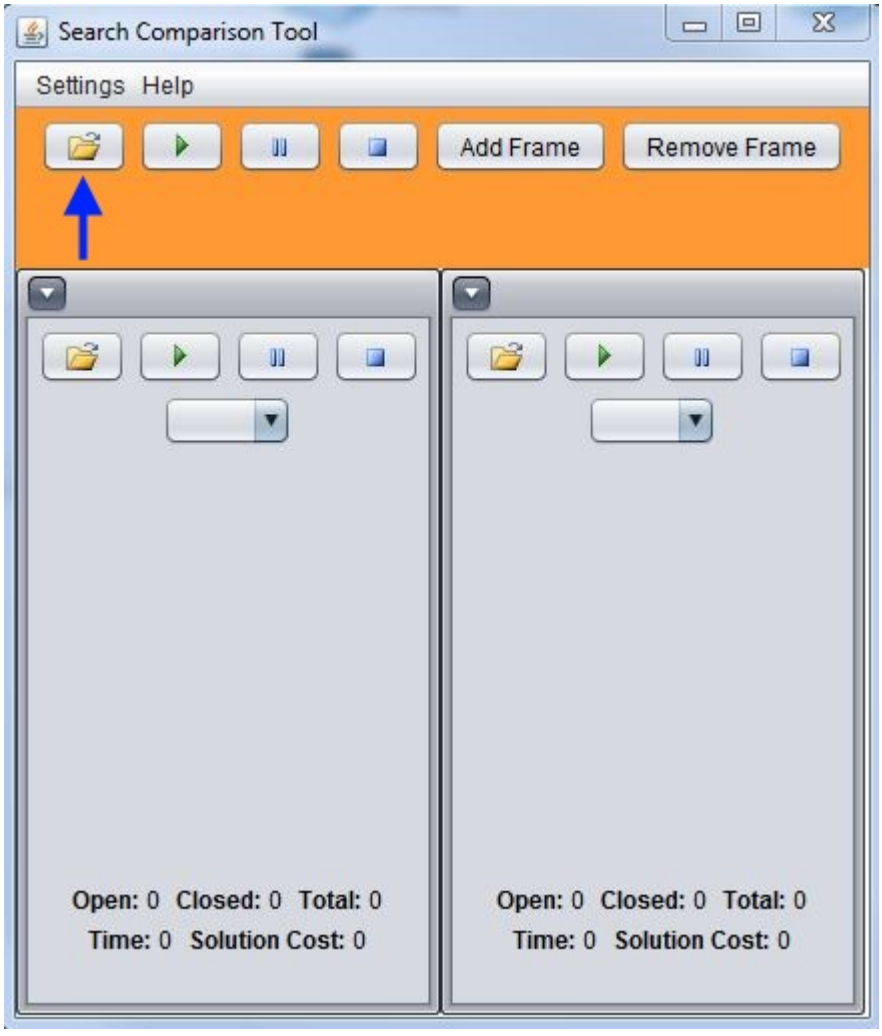
Loading an image

There are two ways to load images in the Search Algorithm App.

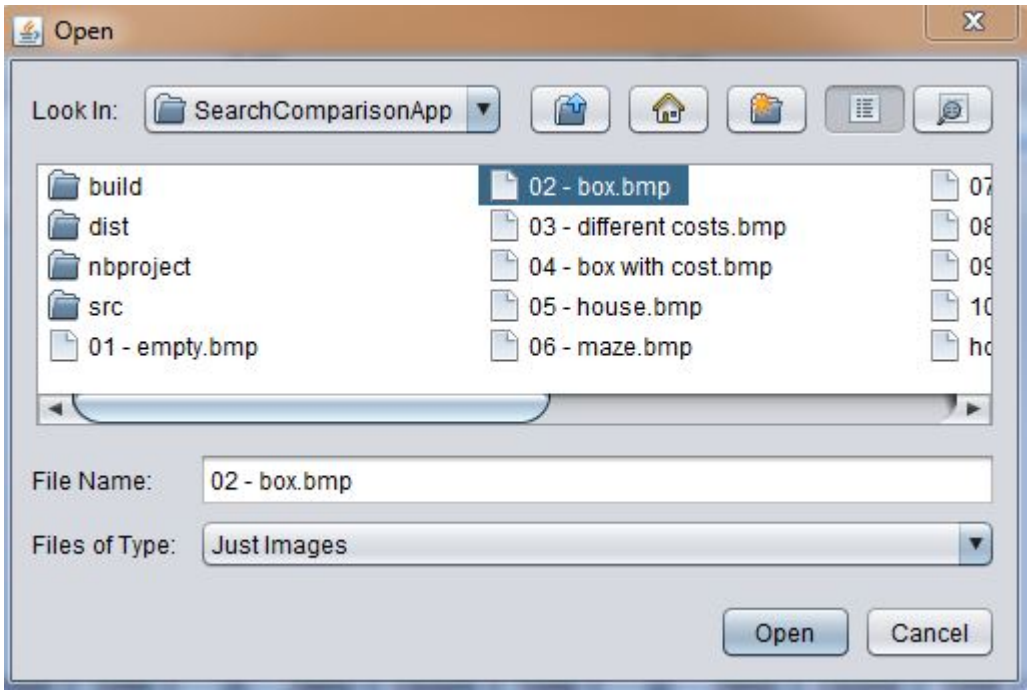
1. Loading an image into all the open search frames
2. Loading an image into an individual search frame

Loading an image into all the open search frames

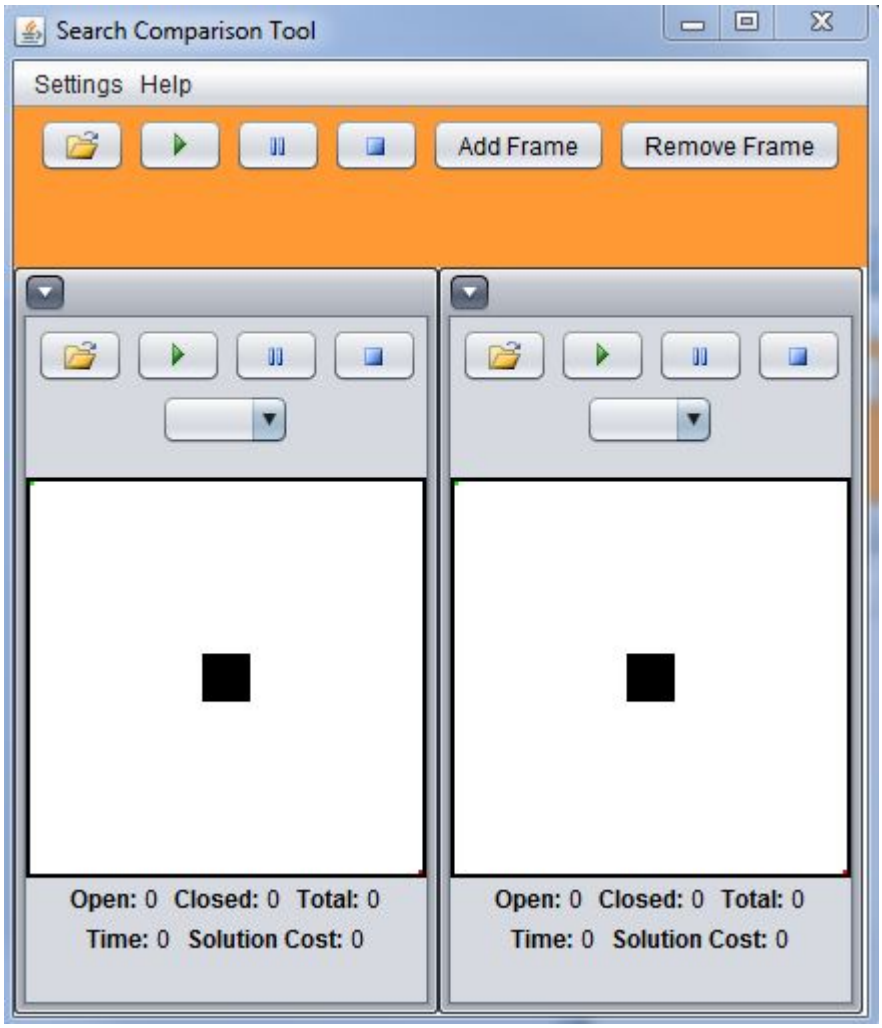
The first way is to load an image into all open search frames at once. To do this click on the button pointed to below by the blue arrow.



You will then see the following file browsing window. By default the window opens to the same directory as the program executable jar file so generally the images will be in this same directory. Select the image you want to load and then click the Open button.



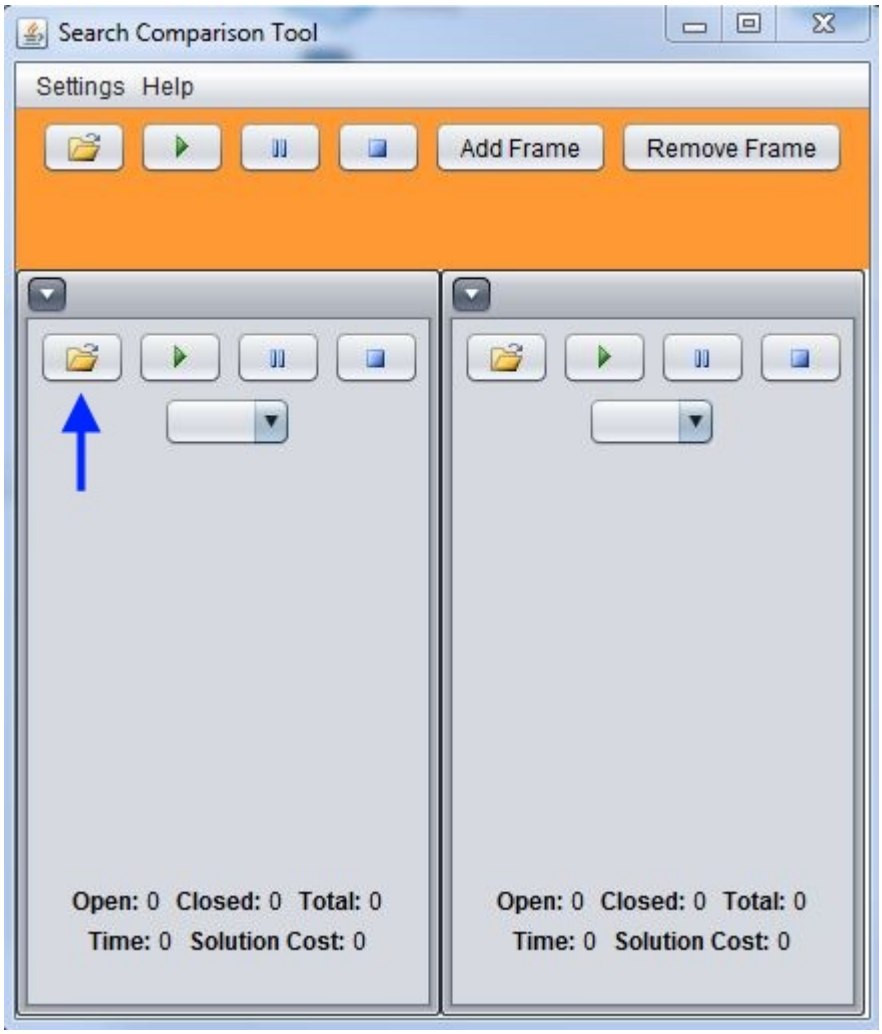
You will now see that all of the search frames have an image loaded inside them.



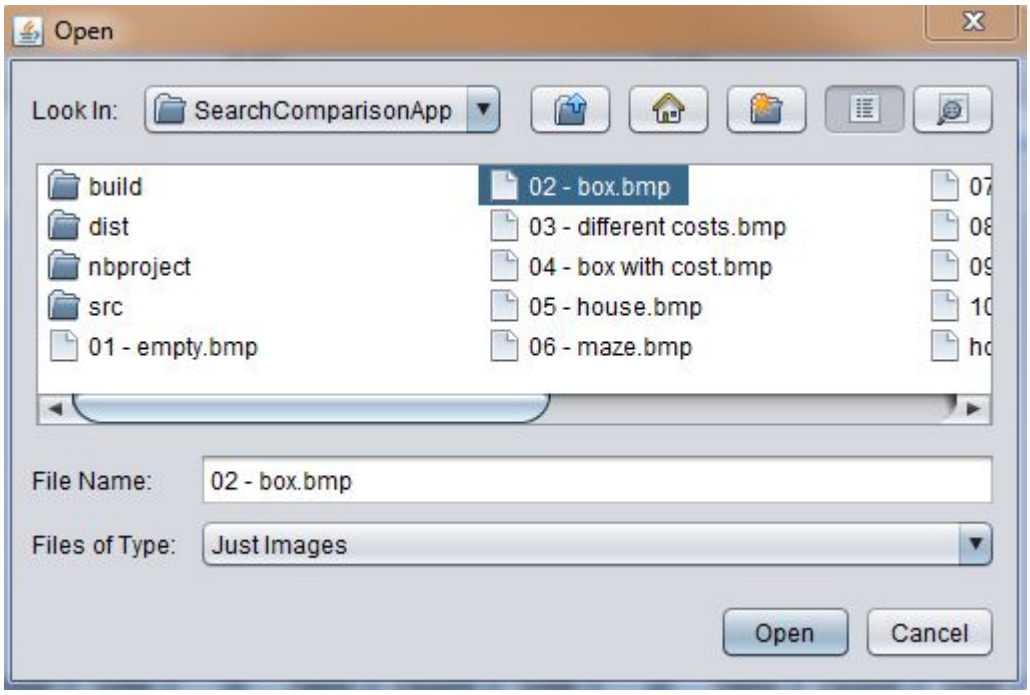
Now that you have an image all ready to be searched it's time to set the starting / ending points of the search

Loading an image into an individual search frame

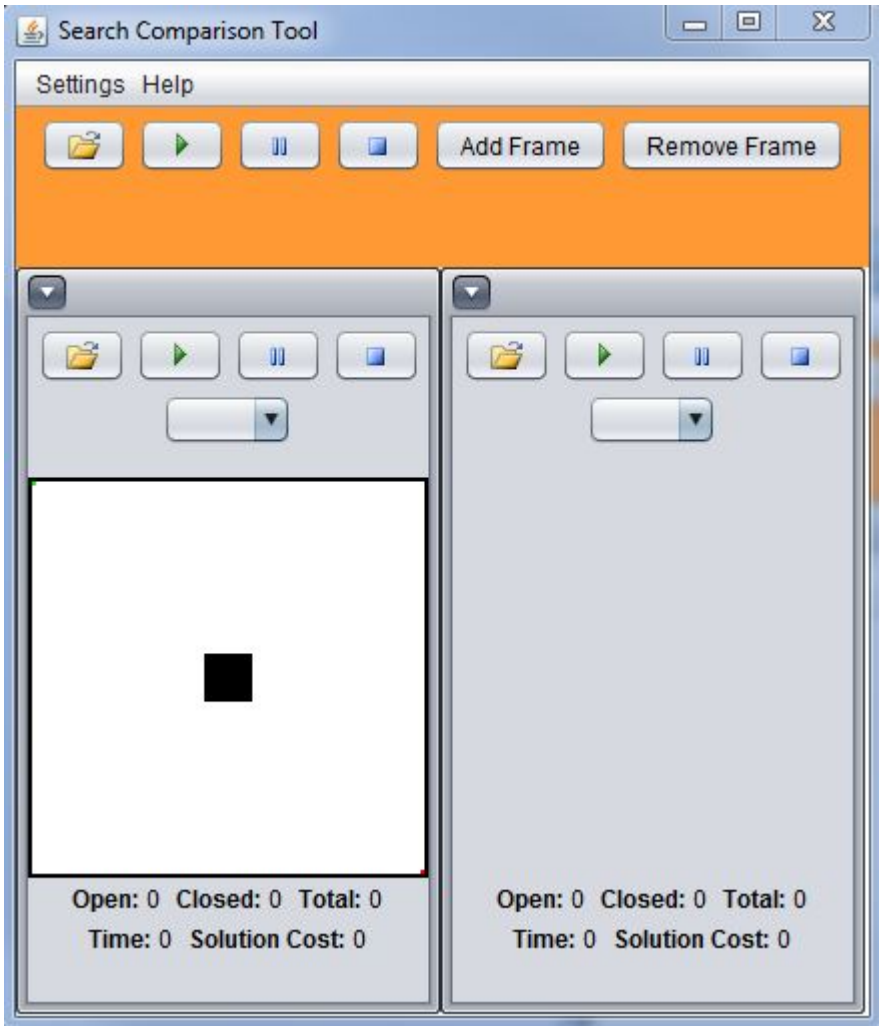
If you want to have different images in each search frame then you can load an image into an individual frame by clicking the button pointed to below by the blue arrow.



You will then see the following file browsing window. By default the window opens to the same directory as the program executable jar file so generally the images will be in this same directory. Select the image you want to load and then click the Open button.

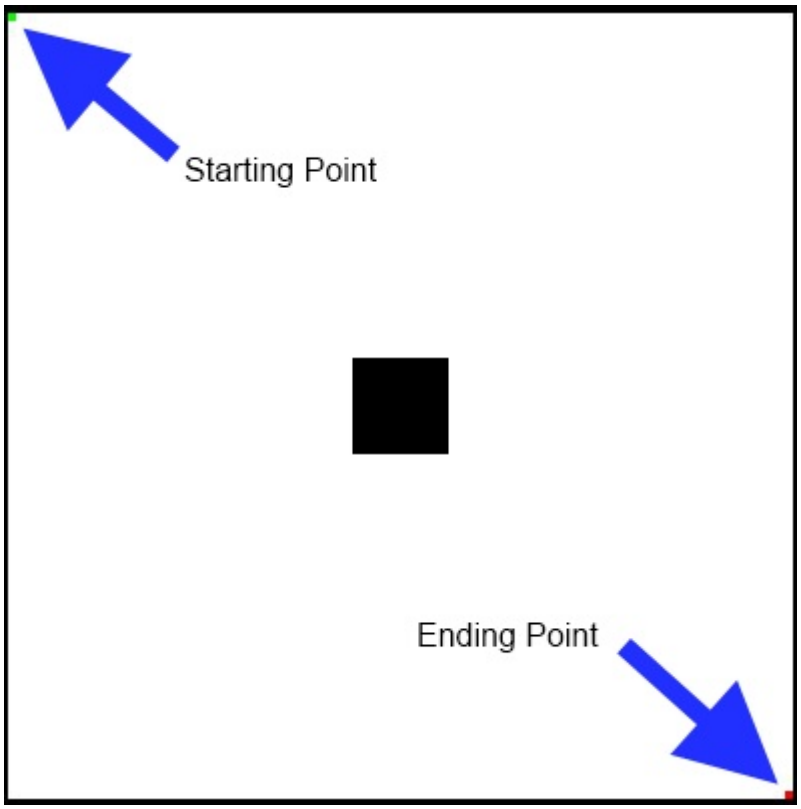


You will now see that the image you selected is only loaded in the search frame who's button you clicked



Now that you have an image all ready to be searched it's time to set the starting / ending points of the search

Setting start / end points



After loading an image the starting and ending points of the search will be in the top left and bottom right corners of the image. The starting point is always denoted by a green square and the ending point is denoted by a red quare as showin in the image above.

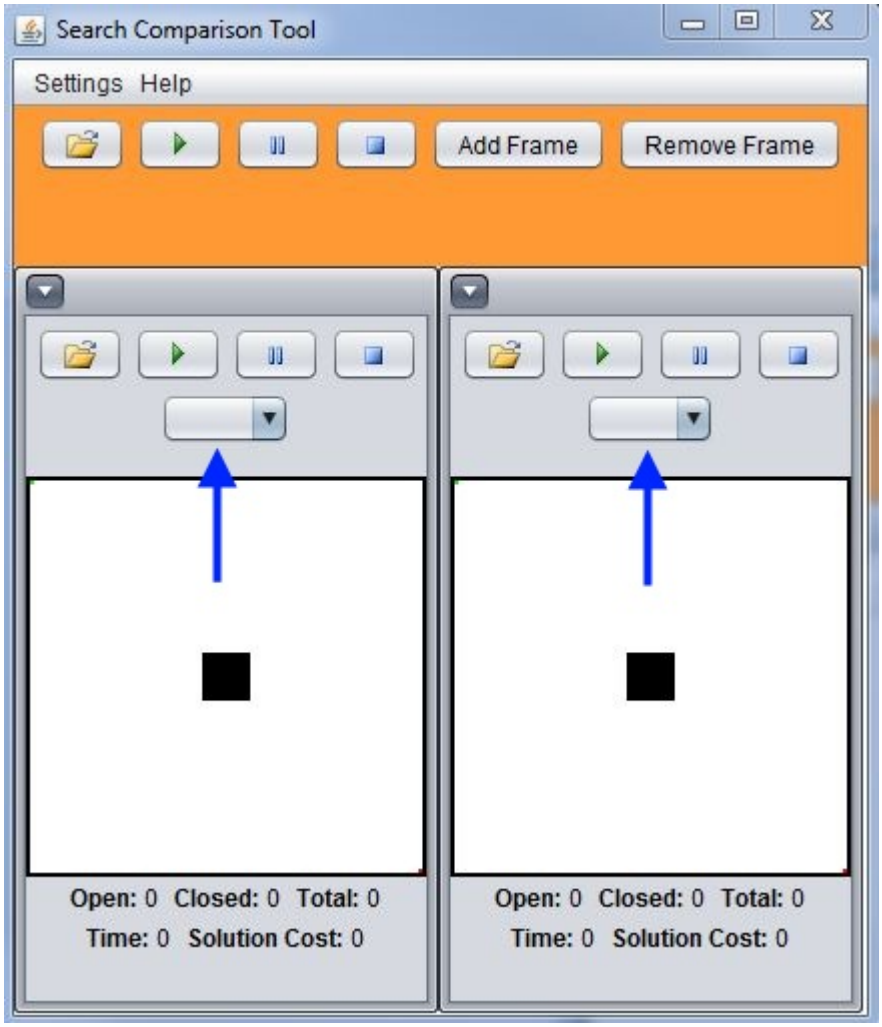
To change the starting point simply move your mouse cursor over the image and quickly left click on the location you want the search to start at. If successful you will now see the green square moved to the point you just clicked on.

To change the ending point simply move your cursor over the image where you want the ending point to be and push the left mouse button down for a 2 seconds and release it. If successful you will now see the red square moved to the point you just clicked on.

Now that your starting / ending points have been set it's time to select a search algorithm

Selecting a search algorithm

To select a search algorithm click on the combo boxes denoted below by the blue arrows and select one of the options. Each Search Frame will need an algorithm selected before a search can be run in it.



An explanation of each search algorithm can be found here

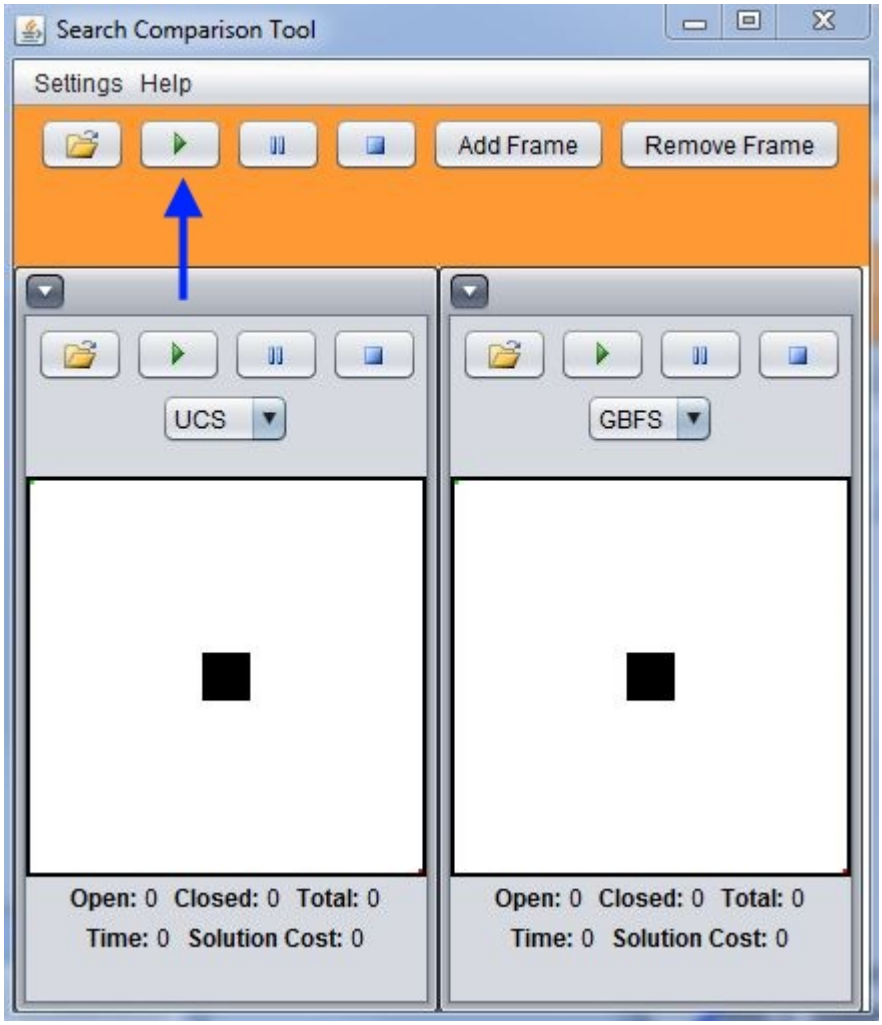
Now that everything is ready go to it's time to start the search!

Starting a search

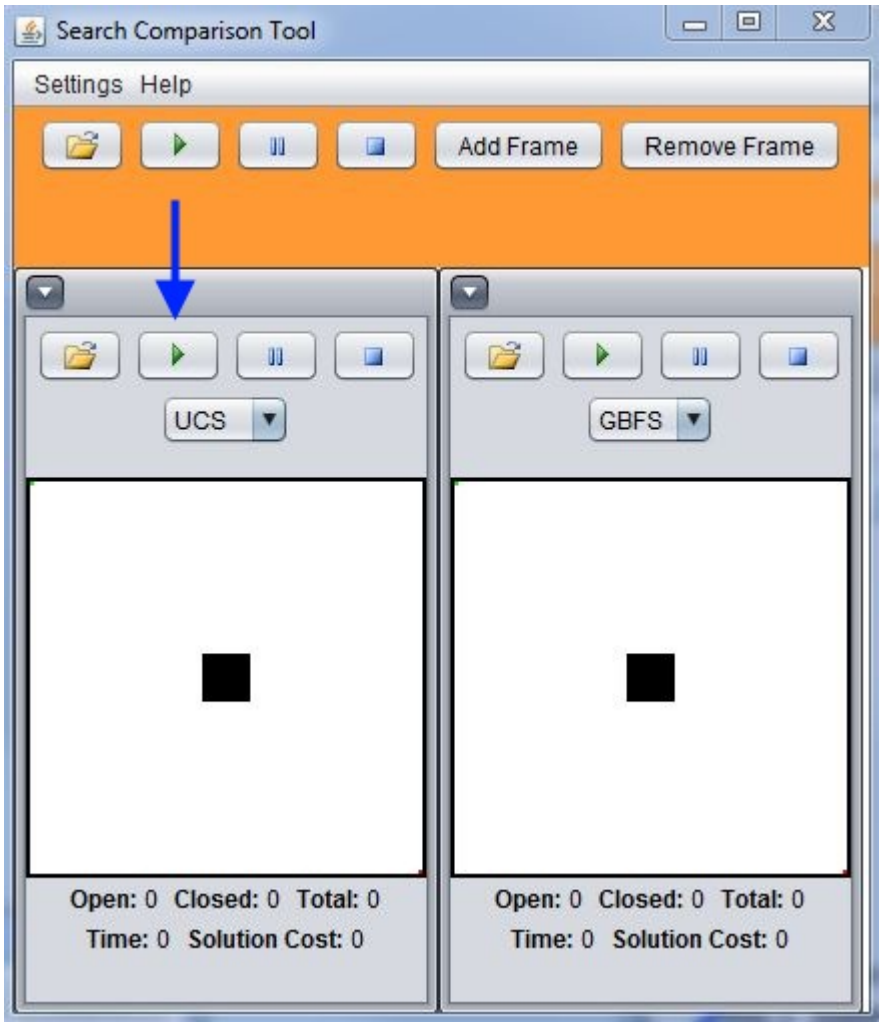
Once an image is loaded and an algorithm is selected you are all ready to run your searches. You have two options:

1. Start searches in all of the search frames that have an image loaded / algorithm selected
2. Start a search in an individual frame

To start all of the searches at once just click the "Play" button denoted below by the blue arrow.



To start a search in any one of the search frames just click on the "Play" button within that frame as denoted below by the blue arro.


Search Comparison Tool
Settings Help
Add Frame
Remove Frame
UCS
GBFS
Open: 0 Closed: 0 Total: 0
Time: 0 Solution Cost: 0
Open: 0 Closed: 0 Total: 0
Time: 0 Solution Cost: 0

Search Process

All of the search algorithms follow the same process in the context of how they work in this program.

When a search is started the first thing that happens is a node is created from the starting point of the image and added to the open list. The search then loops through the following steps until the end node is found or until the open list is empty.

1. Get the first node in the open list

This node will be different depending on the open list type that is used

- Queue - The "oldest" node in the open list. Used for Breadth First Search
- Stack - The "newest" node in the open list. Used for Depth First Search
- Priority Queue - The node in the open list with the lowest cost. How this cost is calculated for nodes will depend on the algorithm used. This is used with Uniform Cost Search, A*, and Greedy Best First Search

Note: This node from the open list will hereafter be referred to as the current node

2. Check to see if the current node is the end node

Nodes are checked for equality by taking the x and y coordiates associated with each node and multiplying them by two prime numbers (3 for x and 5 for y). The resulting values are then compared and if they are equal then the nodes represent the same location on the image. If the node taken from the open list is equal to the end node then the search is terminated.

3. Get the nodes adjacent to the current node

The image pixels are examined in the following order with the green block representing the current node.

1	4	6
2		7
3	5	8

As each pixel is examined the first thing that happens is that the grayscale value of the pixel is checked to see if it is black. Since black pixels in the images represent impassible objects, if the pixel being examined is black no node is created for it and process skips to the next pixel.

If the pixel being examined is NOT black then a node representing this pixel is created and we move onto the next pixel. A node is essentially just a container with the following information:

- The image coordinates of the pixel
- A link to the the nodes parent (the node represented as the green block in this case) which is used to calculate the solution path

- The terrain cost of the pixel (g cost) which is based off the the grayscale value of the pixel (the darker the pixel the higher the terrain cost). This is always calculated even if the search algorithm doesn't make use of this information because we need it to calculate the cost of the solution path.
- The heuristic cost (h cost) - This may or may not be used depending on the search algorithm
- The total cost (f cost) - This may or may not be used depending on the search algorithm

4. For each adjacent node do the following

A) Check to see if the node is in the closed list

If the node is in the closed list then it has already been examined so we can skip to the next adjacent node

B) Set the nodes costs (g, f, h) according to the search algorithm currently in use

Consult the search algorithms to see specifics for each algorithm

C) Add the node to the open list

What happens in this step is dependent on what type of open list is being used by the current search algorithm

- Stack or Queue - Check to see if the node is already in the open list. If it is then skip to the next adjacent node. If it is not then add it to the open list.
- Priority Queue - Check to see if the node is already in the open list. If it is then compare the costs of the two nodes and if the adjacent node we are currently on has a lower cost than the node already in the priority queue, remove the node in the priority queue and add the adjacent node. If it is not then add it to the open list.

5. Add the current node to the closed list

This is necessary for all the algorithms because of the fact that there are many paths to each node and all of the nodes around the current node are examined. If there was no closed list then then there would be paths of infinite lengths.

Search Algorithms

BFS - Breadth First Search

DFS - Depth First Search

UCS - Uniform Cost Search

GBFS - Greedy Best First Search

A*

Breadth First Search

An uninformed (does not use a heuristic) algorithm that searches nodes in *increasing* order of their distance from the starting node. In order to get this behavior a Queue is used for the open list.

To illustrate how Breadth First Search works with respect to this program, a 5 pixel x 5 pixel image is modeled by the table below. For the purposes of the example the middle pixel is the point where the search starts.

The numbers in the table represent the order in which the nodes are taken off the open list and evaluated. The order in which the nodes are added to the open list is dependent on the order in which the adjacent nodes are evaluated (see Search Process) and is not really important.

9	12	13	18	20
10	1	4	6	21
11	2	Start Node	7	22
14	3	5	8	23
15	16	17	19	24

More detailed information about Breadth First Search can be found here on Wikipedia

Depth First Search

An uninformed (does not use a heuristic) algorithm that searches nodes in *decreasing* order of their distance from the starting node. In order to get this behavior a Stack is used for the open list.

To illustrate how Depth First Search works with respect to this program, a 5 pixel x 5 pixel image is modeled by the table below. For the purposes of the example the middle pixel is the point where the search starts.

The numbers in the table represent the order in which the nodes are taken off the open list and evaluated. The order in which the nodes are added to the open list is dependent on the order in which the adjacent nodes are evaluated (see Search Process) and is not really important.

10	9	8	7	6
11	24	21	19	5
12	23	Start Node	18	4
13	22	20	1	3
14	15	16	17	2

More detailed information about Depth First Search can be found here on Wikipedia

Uniform Cost Search

An uninformed (does not use a heuristic) algorithm that searches nodes in increasing order of their cumulative path cost. Uniform Cost Search will produce an optimal solution because all nodes with a cumulative path cost less than or equal to the solution node are evaluated. Because the nodes are sorted a Priority Queue is used for the open list.

To illustrate how Uniform Cost Search works with respect to this program, a 5 pixel x 5 pixel image is modeled by the table below. The path cost of each node is given by its background color according to the legend. The lowest cumulative path cost to each node is denoted by it's numeric value.

A, 2	F, 2	K, 2	P, 5	S, 3
B, 1	G, 2	L, 1	Q, 2	T, 3
C, 1	H, 0 (Start)	M, 1		U, 3
D, 4	I, 1	N, 4		V, 4
E, 2	J, 2	O, 4	R, 8	W, 5 (End)

Color	Cost
	1
	2
	3
	4
	Infinite (Wall)

The step by step progress of the search is given below

Expanded Node	Open List	Closed List
H	B=1, C=1, I=1, L=1, M=1, G=2, D=4, N=4	H
B	C=1, I=1, L=1, M=1, A=2, F=2, G=2, D=4, N=4	B, H
C	I=1, L=1, M=1, A=2, F=2, G=2, D=4, N=4	B, C, H
I	L=1, M=1, A=2, E=2, F=2, G=2, J=2, D=4, N=4, O=4	B, C, H, I
L	M=1, A=2, E=2, F=2, G=2, K=2, J=2, Q=2, D=4, N=4, O=4, P=5	B, C, H, I, L
M	A=2, E=2, F=2, G=2, K=2, J=2, Q=2, D=4, N=4, O=4, P=5	B, C, H, I, L, M
A	E=2, F=2, G=2, K=2, J=2, Q=2, D=4, N=4, O=4, P=5	A, B, C, H, I, L, M
E	F=2, G=2, K=2, J=2, Q=2, D=4, N=4, O=4, P=5	A, B, C, E, H, I, L, M
F	G=2, K=2, J=2, Q=2, D=4, N=4, O=4, P=5	A, B, C, E, F, H, I, L, M

G	K=2, J=2, Q=2, D=4, N=4, O=4, P=5	A, B, C, E, F, G, H, I, L, M
K	J=2, Q=2 D=4, N=4, O=4, P=5	A, B, C, E, F, G, H, I, K, L, M
J	Q=2, D=4, N=4, O=4, P=5	A, B, C, E, F, G, H, I, J, K, L, M
Q	S=3, T=3, U=3, D=4, N=4, O=4, P=5	A, B, C, E, F, G, H, I, J, K, L, M, Q
S	T=3, U=3, D=4, N=4, O=4, P=5	A, B, C, E, F, G, H, I, J, K, L, M, Q, S
T	U=3, D=4, N=4, O=4, P=5	A, B, C, E, F, G, H, I, J, K, L, M, Q, S, T
U	D=4, N=4, O=4, V=4, P=5	A, B, C, E, F, G, H, I, J, K, L, M, Q, S, T, U
D	N=4, O=4, V=4, P=5	A, B, C, D, E, F, G, H, I, J, K, L, M, Q, S, T, U
N	O=4, V=4, P=5, R=8	A, B, C, D, E, F, G, H, I, J, K, L, M, N, Q, S, T, U
O	V=4, P=5, R=8	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, Q, S, T, U
V	P=5, W=5, R=8	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, Q, S, T, U, V
P	W=5, R=8	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, S, T, U, V
W		

Once the goal node is found it's parent node is retrieved and that nodes parent node is retrieved and so on until the start node is found. This constitutes the solution path.

The results of our search are as follows:

Nodes Expanded	Solution Path	Solution Cost
22	H -> L -> Q-> U -> V -> W	5

More detailed information can be found here on Wikipedia

Greedy Best First Search

An informed (uses a heuristic) algorithm that searches nodes in increasing order of their heuristic cost. Because the nodes are sorted a Priority Queue is used for the open list.

To illustrate how Greedy Best First Search works with respect to this program, a 5 pixel x 5 pixel image is modeled by the table below. The path cost of each node is given by its background color according to the legend. The lowest cumulative path cost to each node is denoted by the first number and the heuristic cost of each node is denoted by the second number. The heuristic used is straight line distance which is calculated using the Pythagorean theorem and rounding to the nearest tenth.

A, 2, 5.7	F, 2, 5	K, 2, 4.5	P, 5, 4.1	S, 3, 4
B, 1, 5	G, 2, 4.2	L, 1, 3.6	Q, 2, 3.2	T, 3, 3
C, 1, 4.5	H, 0, 0	M, 1, 2.8		U, 3, 2
D, 4, 4.1	I, 1, 3.2	N, 4, 2.3		V, 4, 1
E, 2, 4	J, 2, 3	O, 4, 2	R, 8, 1	W, 5, 0

Color	Cost
	1
	2
	3
	4
	Infinite (Wall)

The step by step progress of the search is given below

Expanded Node	Open List	Closed List
H	N=2.3, M=2.8, I=3.2, L=3.6, D=4.1, G=4.2, C=4.5, B=5	H
N	R=1, O=2, M=2.8, I=3.2, J=3, L=3.6, D=4.1, G=4.2, C=4.5, B=5	H, N
R	W=0, V=1, O=2, M=2.8, I=3.2, J=3, L=3.6, D=4.1, G=4.2, C=4.5, B=5	H, N, R
W		

Once the goal node is found it's parent node is retrieved and that nodes parent node is retrieved and so on until the start node is found. This constitutes the solution path.

The results of our search are as follows:

Nodes Expanded	Solution Path	Solution Cost
4	H -> N -> R -> W	9

More detailed information can be found here on Wikipedia

A*

An informed (uses a heuristic) algorithm that searches nodes in increasing order of their total cost (cumulative path cost + heuristic cost). A* will produce an optimal solution as long as the heuristic used is admissible because all nodes with a total cost less than or equal to the end node are evaluated. Because the nodes are sorted a Priority Queue is used for the open list.

To illustrate how A* works with respect to this program, a 5 pixel x 5 pixel image is modeled by the table below. The path cost of each node is given by its background color according to the legend. The lowest cumulative path cost to each node is denoted by the first number, heuristic cost of each node is denoted by the second number, and the total cost is denoted by the third number. The heuristic used is straight line distance which is calculated using the Pythagorean theorem and rounding to the nearest tenth.

A, 2, 5.7, 7.7	F, 2, 5, 7	K, 2, 4.5, 6.5	P, 5, 4.1, 9.1	S, 3, 4, 7
B, 1, 5, 6	G, 2, 4.2, 6.2	L, 1, 3.6, 4.6	Q, 2, 3.2, 5.2	T, 3, 3, 6
C, 1, 4.5, 5.5	H, 0, 0, 0	M, 1, 2.8, 3.8		U, 3, 2, 5
D, 4, 4.1, 8.1	I, 1, 3.2, 4.2	N, 4, 2.3, 6.3		V, 4, 1, 5
E, 2, 4, 6	J, 2, 3, 5	O, 4, 2, 6	R, 8, 1	W, 5, 0, 5

Color	Cost
	1
	2
	3
	4
	Infinite (Wall)

The step by step progress of the search is given below

Expanded Node	Open List	Closed List
H	M=3.8, I=4.2, L=4.6, C=5.5, B=6, G=6.2, N=6.3, D=8.1	H
M	I=4.2, L=4.6, Q=5.2, C=5.5, B=6, G=6.2, N=6.3, D=8.1	H, M
I	L=4.6, J=5, Q=5.2, C=5.5, B=6, E=6, O=6, G=6.2, N=6.3, D=8.1	H, I, M
L	J=5, Q=5.2, C=5.5, B=6, E=6, O=6, G=6.2, K=6.5, N=6.3, F=7, D=8.1, P=9.1	H, I, L, M
J	Q=5.2, C=5.5, B=6, E=6, O=6, G=6.2, K=6.5, N=6.3, F=7, D=8.1, P=9.1	H, I, J, L, M
Q	U=5, C=5.5, B=6, E=6, O=6, T=6, G=6.2, K=6.5, N=6.3, F=7, S=7, D=8.1, P=9.1	H, I, J, L, M, Q
U	V=5, C=5.5, B=6, E=6, O=6, T=6, G=6.2, K=6.5, N=6.3, F=7, S=7, D=8.1, P=9.1	H, I, J, L, M, Q, U

V	W=5, C=5.5, B=6, E=6, O=6, T=6, G=6.2, K=6.5, N=6.3, F=7, S=7, D=8.1, P=9.1	H, I, J, L, M, Q, U, V
W		

Once the goal node is found it's parent node is retrieved and that nodes parent node is retrieved and so on until the start node is found. This constitutes the solution path.

The results of our search are as follows:

Nodes Expanded	Solution Path	Solution Cost
9	H -> M-> Q -> U -> V -> W	5

More detailed information can be found here on Wikipedia