
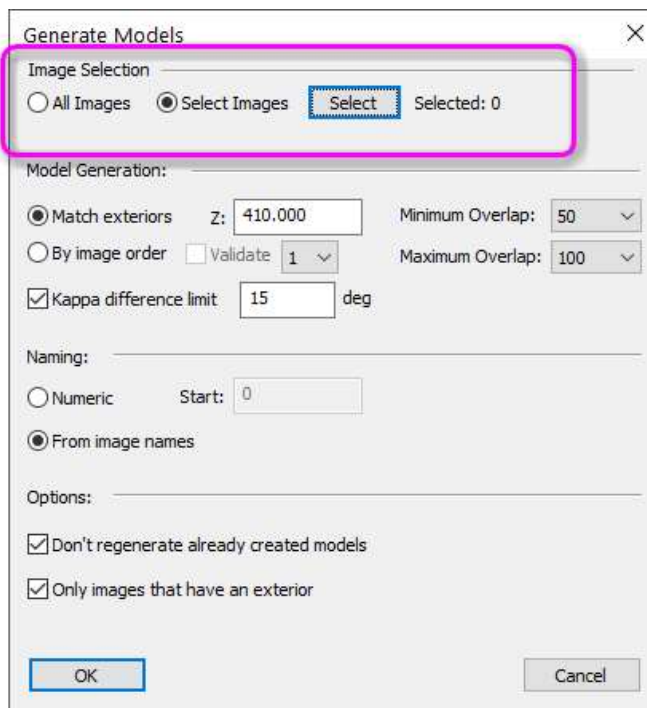
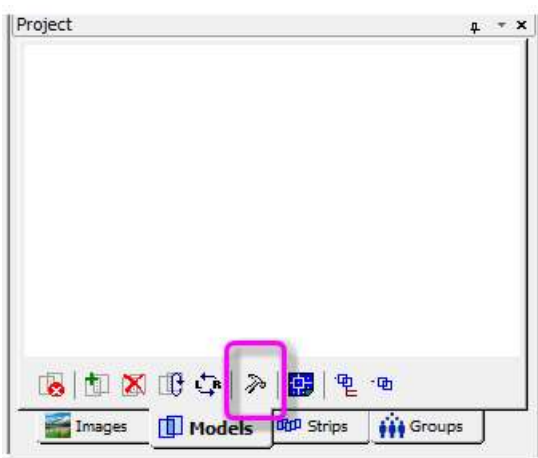


Starting in DAT/EM Summit Evolution version 8.0, there is a new optional filter to select only certain images to be used for automatic model generation. The filter contains “Regular Expressions”, which are a set of searching standards. If you are not familiar with using Regular Expressions, use this quick tutorial to help you get started.

Regular Expressions can be very detailed and they have many options. Here, we will get you started with using simple expressions in Summit’s model generator to help filter images by characters or strings in their file names. We will also provide links to resources so you can learn about more Regular Expression components.

The new filter is here: **Summit > Project Window > Models tab >  > Image Selection area > Select:**



Things to keep in mind:

- The wildcard * can be used in Regular Expressions, but it does not work quite the same way that you might expect based on its use in other applications. The proper way to use it is .* and this is shown in the examples below.

- Summit is using a *Boost::regex_search* that will allow a partial match. Some other types of Regular Expressions require an exact match, but partial matches will work better for selecting images by parts of their file names.
- Refer to <https://www.autohotkey.com/docs/misc/RegEx-QuickRef.htm> AND https://en.wikipedia.org/wiki/Regular_expression for how Regular Expressions normally work. Since we allow partial matches, our matches will work slightly differently from what might be expected.

Let's say we have a list of images from a multi-head (multi-directional) camera such as the Vexcel Osprey 4.1. There may be hundreds of images from each camera direction in a project, but our example has a small subset:

0022-RGB-Left.smti	0166-RGB-Fwd.smti
0023-RGB-Left.smti	0167-RGB-Fwd.smti
0056-RGB-Right.smti	0174-RGB-Color.smti
0057-RGB-Right.smti	0175-RGB-Color.smti
0124-RGB-Fwd.smti	0182-RGB-Bwd.smti
0125-RGB-Fwd.smti	0183-RGB-Bwd.smti
0132-RGB-Color.smti	0250-RGB-Right.smti
0133-RGB-Color.smti	0251-RGB-Right.smti
0140-RGB-Bwd.smti	0284-RGB-Left.smti
0141-RGB-Bwd.smti	0285-RGB-Left.smti

We want to make models only from matching camera directions, as indicated by the highlight colors above.

You might have something different, such as multiple years of imagery that used a different naming convention. If there is some unique part of the file name string that you can detect by reading them, then you can build a Regular Expression that will find just those images.

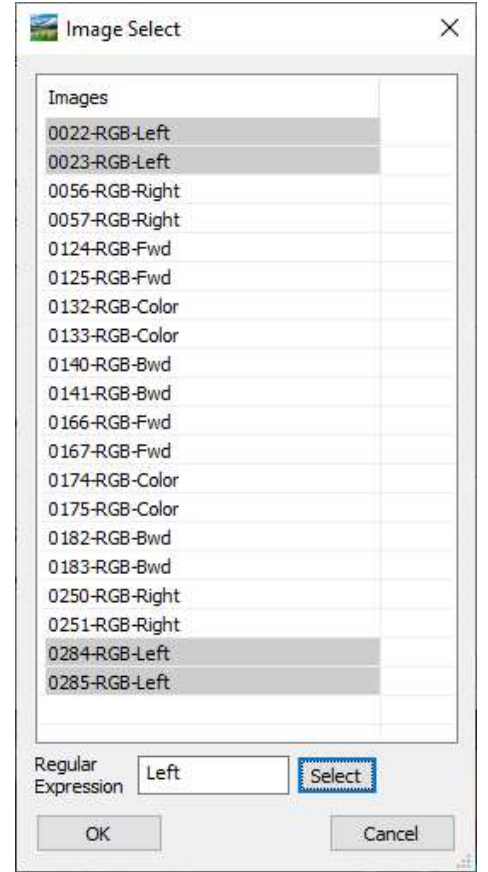
The following are some example Regular Expressions that will help find these image sets using partial matches:

Left

This works to find the images with 'Left' somewhere in their name.

'Left' works, because we allowed partial matches. We found a piece of the model name that matches the expression in four of the images in our example, and they are now selected and highlighted in gray.

If we select **OK** now, only the highlighted images will be used for model generation. Only the highlighted images will be sent on to the next set of filters, which appear back in the main model generation dialog, such as the % overlap and kappa filters.



028.*Left

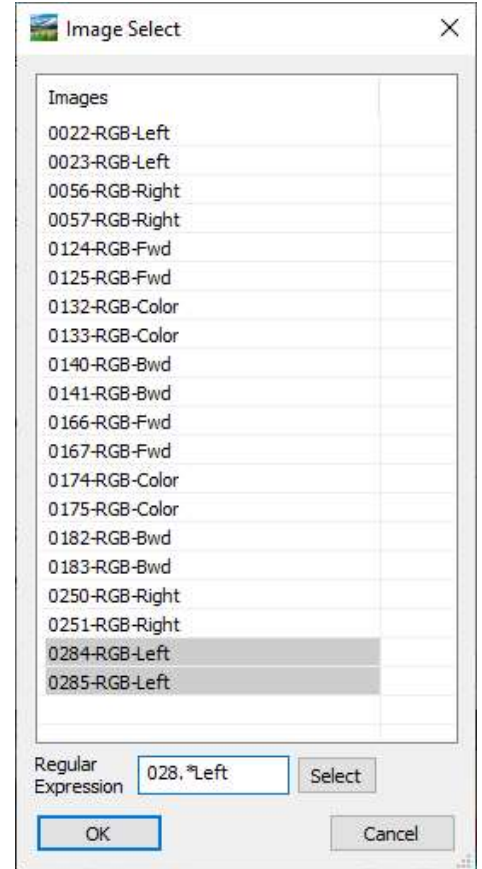
Notice that fewer items are now selected than when you used simply 'Left'. If we break down this expression, the different parts do this:

028 : This tells it to start by matching exactly the characters '028'.

.* : Next take any number of characters and match to them. You can also think of this as skipping any number of any characters. (Note the difference between the more common use of the * wildcard in other applications. In Regular Expressions, * can't be used by itself. It must be listed as .* in the expression.)

Left : This tells it to match the characters 'Left' anywhere in the remaining part of the string.

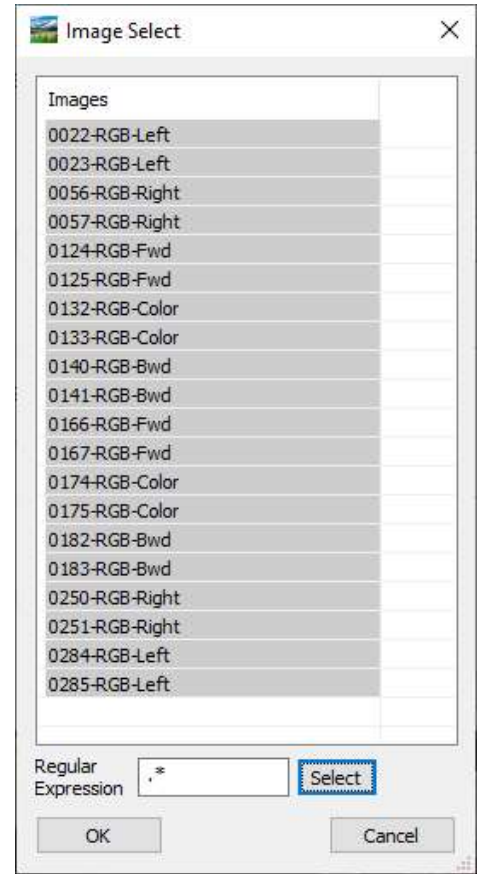
Now only two of the left-facing camera images are selected, because the other two that contained 'Left' don't contain '028' at the beginning of their name.



.*

We include this as an example to further show the effects of the .* wildcard. If you use .* as the entire regular expression, it will find all images.

This obviously won't be useful to you by itself, but it can be useful when combined with other expression components, such as the example immediately above.



(1 dot)

May be repeated, such as 6 of them:

.....

(6 dots)

Each dot represents exactly 1 character, which may be any character. For example, 6 dots means to allow any characters for exactly 6 character positions.

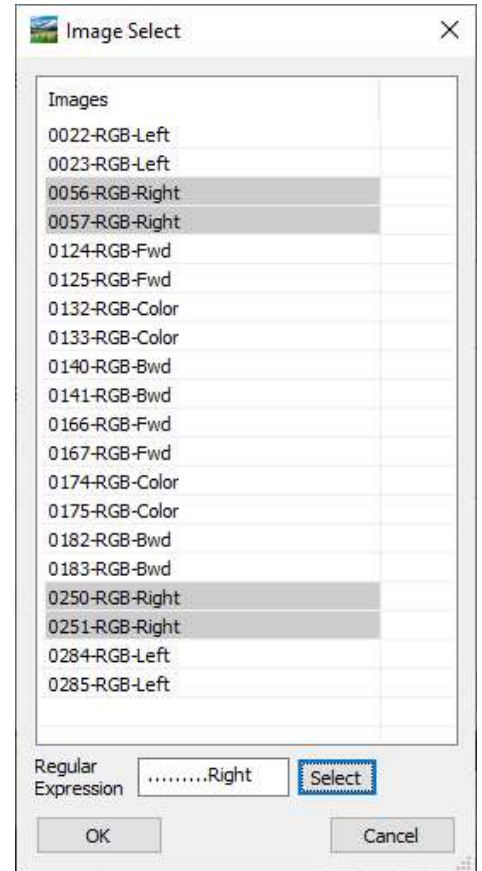
Now let's turn this into a more useful expression:

.....**Right**

(9 dots)

This starts with exactly 9 dots, so it will allow any characters for the first 9 positions, but then 'Right' must appear somewhere in the rest of the string. 'Right' may start on the 10th character or anywhere after that.

In this case, it gives the same result as if we had entered only **Right** by itself; however, there could be cases of multiple combined image sets where the dots help you exclude images with a common string in different parts of their file names. For example, if another image set had 'Right' only at the beginning of the string, it would exclude those images, because it skipped the first 9 character positions.



...Right

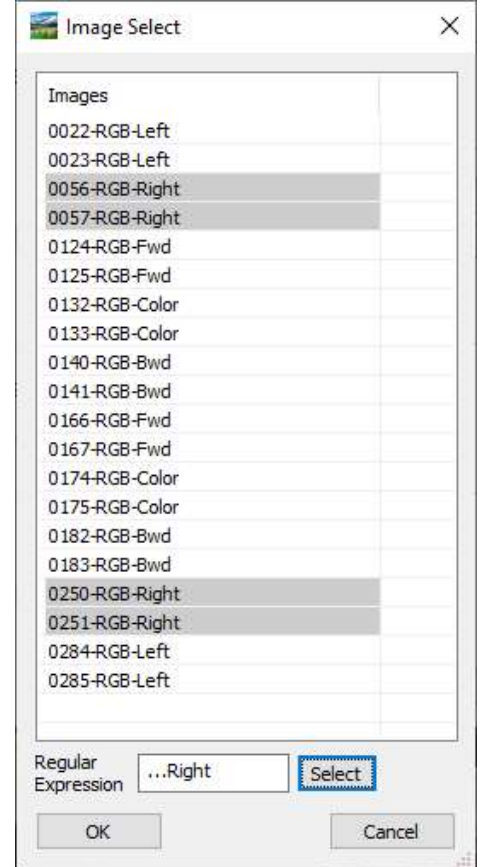
(3 dots)

Why does this give exactly the same results as the example above that used 9 dots? It's because we allowed partial matches.

Really what matched was the part of the string past the first 3 characters. So, it found 'Right' somewhere in '6-RGB-Right', '7-RGB-Right', '0-RGB-Right', and '1-RGB-Right', having skipped the first 3 characters of each of these file names.

When we used the 9 dots in**Right**, it was a much more strict expression that meant 'Right' had to appear somewhere starting in the 10th or higher character positions.

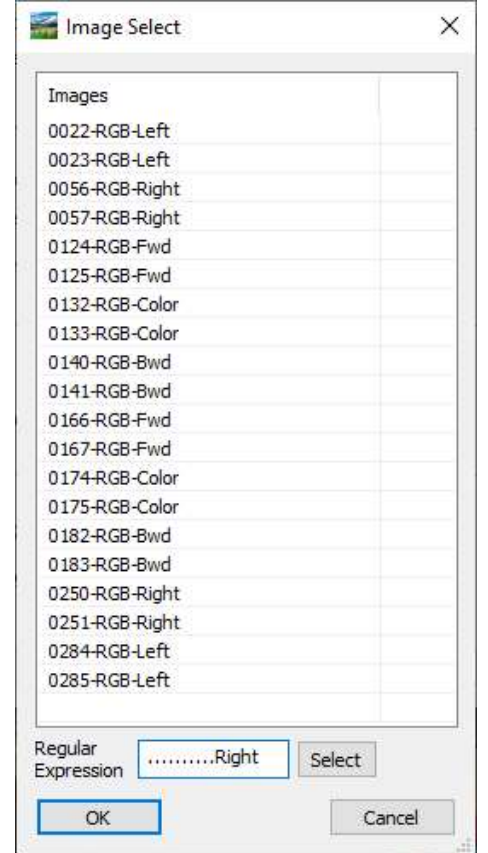
If we had a file name that started with 'Right', such as **Right_07_23_1984.tif**, then an expression starting with 5 dots,**Right**, would miss selecting this image. This is because 'Right' doesn't appear anywhere starting in the 6th or higher character positions. It's searching for 'Right' in '_07_23_1984' and doesn't find it.



.....Right
(10 dots)

We purposely added too many dots just to show what happens.

This time there are 10 dots. It doesn't find anything, because only 'ight' shows up in the 11th and higher part of the string. 'Right' can't be fully found in 'ight', so no images are selected.



...5.*[td]

Now we're getting even more specific with a combination of expression components.

This expression says, in order:

... : Match the first 3 and exactly 3 characters, which can be anything. This can also be thought of as skipping exactly 3 characters at the beginning.

5 : The 4th character must be a '5'. *Note that there are 3 images that have a '5' in the 4th character, but only 2 are selected, so something else in the expression must have ruled out one of the images.*

.* : Next take any number of characters and match to them. You can also think of this as skipping any number of characters.

[td] : The last character must be either a 't' or a 'd'. *This is the part of the expression that rules out the image that has '5' in the 4th character, but ends with an 'r'.*

This is the first time we've introduced an expression component such as **[td]**. You can learn more about how to use the square brackets, **[]**, and even more Regular Expression components in these references:

<https://www.autohotkey.com/docs/misc/RegEx-QuickRef.htm>

and https://en.wikipedia.org/wiki/Regular_expression

