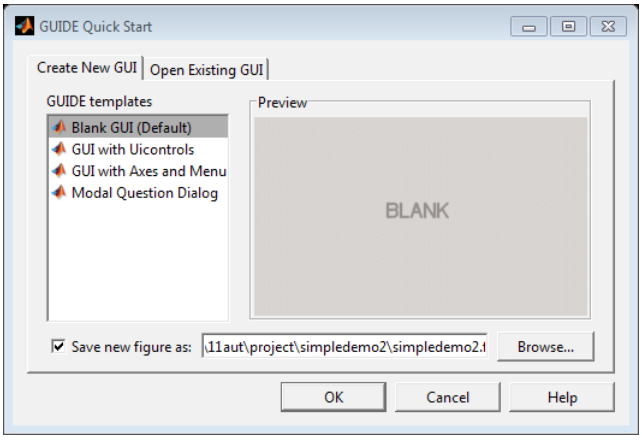
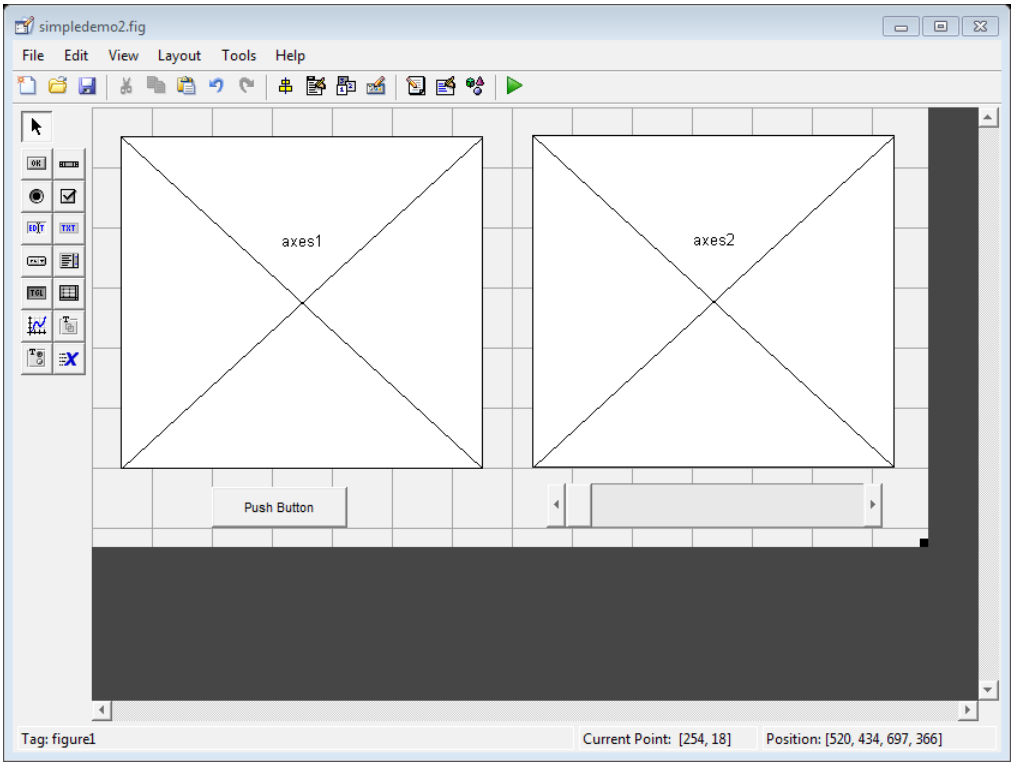
How the simple demo of Gaussian filter is implemented

1. Type `guide` in MATLAB, and select “Blank GUI(Default)” to create a new GUI



2. In the GUI design window, place two “Axes” as the display windows for the original and filtered images. Add a button and a slider to load the image and control the size of the filter.



3. Double-click the push button, its property window will pop up. Change the "String" property to "Load". Double-click the slider, change its "Max" property to 10, and "Min" property to 3, so that the value range of the slider will be between 3 and 10. Also set its "Value" property to 3, since it is the initial value of the slider control and it has to be between the maximun and minimun value.

4. Now add the code for the call back funtion of the Load button. Choose View → M-file Editor, you will see the source code. Find the following function, which is the callback function of the button:

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

Now you can add the following code there to load an image when this button is pressed:

```
global X; % original image
global hAxes1;

% open an image
[FileName,PathName] = uigetfile('*.bmp;*.tif;*.jpg;*.hdf','Select the image file');
FullPathName = [PathName,'\',FileName];
X = imread(FullPathName);

%display the original image
set(gcf, 'CurrentAxes', hAxes1);
imshow(X);

% display the result image
displayResult;
```

By using the `global` declaration the variable `X` which holds the original image data is accessible to all functions, and `hAxes1` is the handle to the first axes. `hAxes1` will be defined in another initialization function later. The `uigetfile()` function opens an open file window and let the user selects an image file to open. The `set()` function set the current axes to be that axes. The `gcf` variable in the `set()` function means the current figure, i.e., the main GUI you are working on. The `displayResult` function is another function used to do the filtering and show the result. I write it in another function because it will be called again every time the user slides the slider and changes the Gaussian filter parameter. It's good practice to write simple, short, independent functions that can be re-used.

Add the code for the function `displayResult`:

```
function displayResult

global X;
global hAxes2;

% get the filter size
hSlider = findobj(gcf, 'Tag', 'slider1');
sz = get(hSlider,'Value');

% filtering
H = fspecial('gaussian', round(sz), 0.5 * round(sz));
Y = imfilter(X, H);

% show the result
set(gcf, 'CurrentAxes', hAxes2);
imshow(Y);
return;
```

Again, `X` is the global variable that holds the original image data. `hAxes2` is the handle to the second axes where we want to show the result image. `findobj()` finds the handle to the slider object, and `get()` retrieves the value associated with the slider. This value is the size of the filter, which we used to generate the filter kernel `H`, and use the `imfilter()` function to do the filtering. `set()` function sets the current axes to the `hAxes2`, and show the result image on it.

We see everything up to now is set up appropriately except the handles to the two image axes where we want to display the image. This is done by another separate function `reset()`:

```
function reset
global hAxes1;
global hAxes2;

if (isempty(hAxes1))
    hAxes1 = findobj(gcf,'Tag', 'axes1');
end
if (isempty(hAxes2))
    hAxes2 = findobj(gcf,'Tag', 'axes2');
end

set(gcf, 'CurrentAxes', hAxes1);
imshow(1);
set(gcf, 'CurrentAxes', hAxes2);
imshow(1);
return;
```

This function defines the handles as global variables so that they can be referenced from other part of the program. The handles are retrieved by the `findobj()` function. We further use `imshow(1)` to "clean up" the image show area. It's not essential to have the last four lines of code, they are purely for aesthetics purpose. You can try commenting out these 4 lines and see the effect.

Now the reset function shall be put right after the program is loaded, and we want to initialize the handles there so that other parts of the program such as loading and filtering can have correct handles to show the

results. So we add the following two lines of code at the end of `simpledemo2_OpeningFcn()` which is generated automatically:

```
% --- Executes just before simpledemo2 is made visible.
function simpledemo2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to simpledemo2 (see VARARGIN)

% Choose default command line output for simpledemo2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes simpledemo2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
clear all;
reset;
```

So that right after the program is loaded, all previous data is cleaned and the handles to the axes are initialized.

5. With all these works done, the call back function for the slider (i.e., the action the program takes when the user slides the slide bar) is to retrieve the value associated with the slider bar, do the filtering, and show the result in axes2. This is exactly what the `displayResult` function does.

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
displayResult;
```

6. So the program is finished. We can see the key for this kind of simple demo is to have some global variables and handles to hold the image data and change display area. Important functions you will be using a lot are `findobj()`, `set()`, and `get()`. This tutorial shows the way to use the slider control and get some parameter. You can study the code of the edge detection demo to see the use of other controls such as radio button and checkbox. They are fairly straightforward to learn. The key thing is not these controls of GUI and how to use them, they are just tools. The key thing is the image processing techniques. Be creative when you finding your own transforms and implementing the demo.