

A data-science approach to art history

Lior Shamir

Kansas State University

Contents

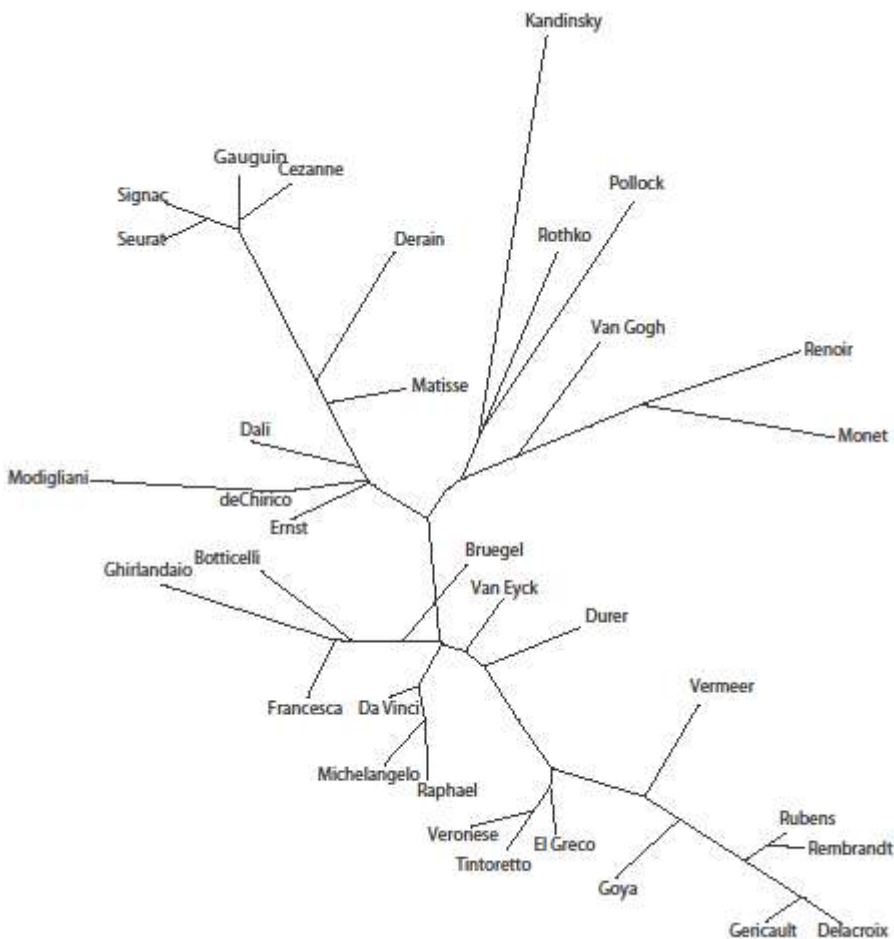
1. Types of experiments.....	2
1.1. Network of similarities between artists.....	2
1.2. What artist had the strongest influence on another artist?	2
1.3. Profile the change in the artistic style of a painter.....	3
1.4. Chronological progression of artistic style.....	3
2. Collecting the data	4
3. Preparing the image files for processing (pre-processing)	9
4. Copying files to the server	14
5. Viewing files on the server.....	22
6. Processing the files on the server	24
7. Copying the files from the server to the laptop.....	27
8. Analyzing the files and creating phylogenies.....	30
8. Enabling algorithms	47
Bibliography	49

1. Types of experiments

There is an unlimited types of experiments that can be designed using this protocol, and experimental design can vary between experiments to explore new questions in art and art history. Here are several examples of experiments that can be done.

1.1. Network of similarities between artists

The experiment profiles similarities between different painters, and produces a network of influential links between different painters. The painters can belong in different schools of art or artistic styles. The following graph shows the results of a previous experiment with 35 western art painters (Shamir & Tarakhovsky, 2012).



1.2. What artist had the strongest influence on another artist?

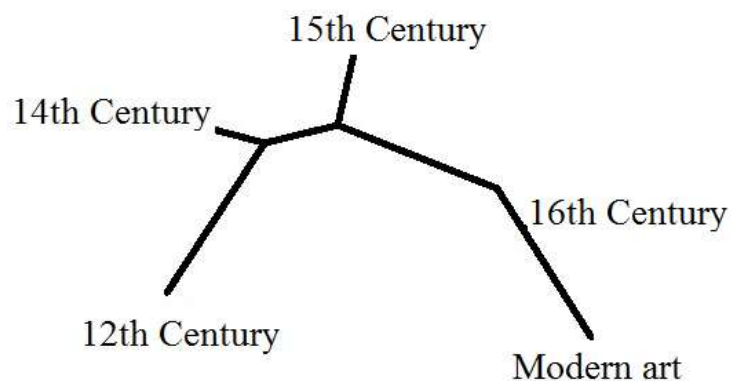
In that kind of experiment paintings of one artist will be collected in addition to paintings of several other artists. The computer will then assist to determine which of the artists is the most similar to the studied painter.

1.3. Profile the change in the artistic style of a painter

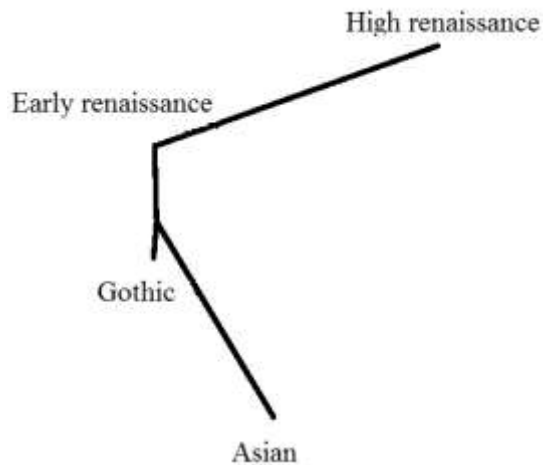
Paintings from different eras in the painter's career and life will be collected. The computer will profile the changes in the artistic style, and the profile can be compared to artistic and biographical data about the painter.

1.4. Chronological progression of artistic style

In the experiment paintings from different eras will be analyzed, and the computer will generate a graph showing a quantitative analysis of the change, identifying periods of rapid changes as well as periods in which the change in artistic style was mild. For example, the following graph was produced by the computer analysis of European art paintings from different centuries.



Another example is the following graph, combining different times and different locations where the paintings were created.



2. Collecting the data

Once the experimental design is completed, and the painter(s) or paintings that will be used for the experiment are known, data should be collected. The data used in all of the experiments are digital paintings. The digital paintings are image files, in most cases JPG file (compressed image files). Each painting image file is also associated with the metadata. The metadata can be the name of the painter, title of the paintings, genre of the paintings, and year in which the painting was created. The metadata is collected and stored based on the need of the experiment. For instance, in the case of experiments 1.1 and 1.2, the only metadata needed for each painting is the name of the painter. For experiment 1.3 the year in which the painting was created is also needed. In experiment 1.4 the required metadata can vary, and can be the school of art, year of the painting, or other metadata based on the experiment.

There are numerous sources for digital paintings.

1. Google art project (<https://www.google.com/culturalinstitute/project/art-project>).
2. Wikimedia (http://commons.wikimedia.org/wiki/Main_Page). Wikimedia also contains many of the Google Art project ([http://commons.wikimedia.org/wiki/Category:Google Art Project works by collection?uselang=en-gb](http://commons.wikimedia.org/wiki/Category:Google_Art_Project_works_by_collection?uselang=en-gb)).
3. The Metropolitan on-line collection (<http://www.metmuseum.org/collection/the-collection-online>).
4. The Louvre museum archive and on-line databases (<http://www.louvre.fr/en/moteur-de-recherche-oeuvres?tab=3#tabs>).
5. Museum of Modern Art (<http://www.moma.org/explore/collection>).
6. Guggenheim Museum collection (<http://www.guggenheim.org/new-york/collections/collection-online>).

7. Art Institute of Chicago (<http://www.artic.edu/aic/collections/>).
8. Tate Gallery collection (<http://www.tate.org.uk/art/>).
9. Image search engines such as Google Images (<http://images.google.com>), Yahoo Images (<http://images.search.yahoo.com/>), Quality Image Search (<http://qualityimagesearch.com/>), Bing Images (<https://www.bing.com/images/>).

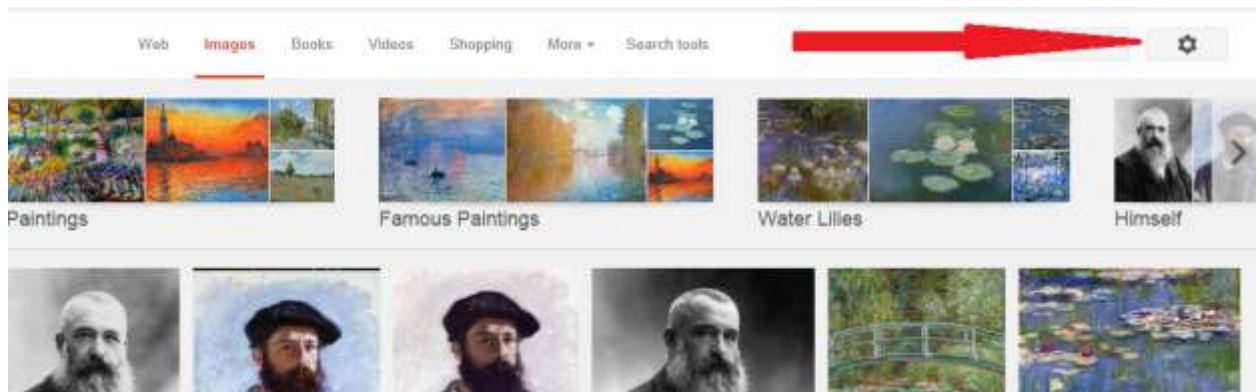
One of the most effective sources from which digital paintings can be obtained is *Google Images* (<http://images.google.com>). *Google Images* is not an on-line art collection and is not designed specifically for searching digital paintings, but as a general comprehensive search engine it can be used to search and download digital paintings. Other search engines such as *Bing Images* can also be used, and will likely find images not found by *Google Images*.

For instance, assuming that we are interested in paintings of Claude Monet, here are the steps for obtaining digital paintings from Google Images.

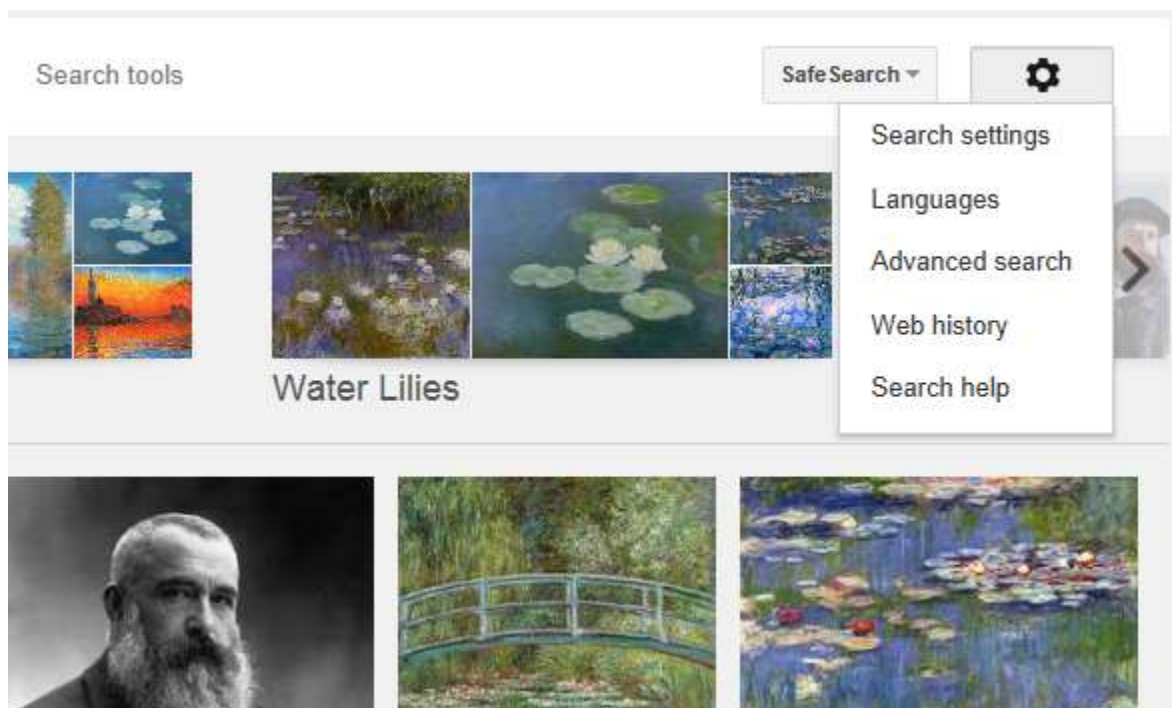
1. Go to Google Images at <http://images.google.com>.
2. Type the name of the painter, in our example “Claude Monet” and click the search button.



3. That will bring a list of thumbnails of Claude Monet images. However, many of these images are too small for effective computer analysis. Therefore, we need to search for images that are larger than 800x600 pixels. To do that, we need to change the search settings by clicking the “advanced search” option at the top right corner of the screen.



4. After clicking the button, choose "Advanced search".



5. In the advanced search, find "image size:", and then select "larger than 800x600".

any of these words: Type OR between all the words you want: trees OR weeds OR grasses

none of these words: Put a minus sign just before words you don't want: -trees -weeds -grass

Then narrow your results by...

image size: Find images in any size you need.

aspect ratio: Specify the shape of images.

colors in image: Find images in your preferred colors.

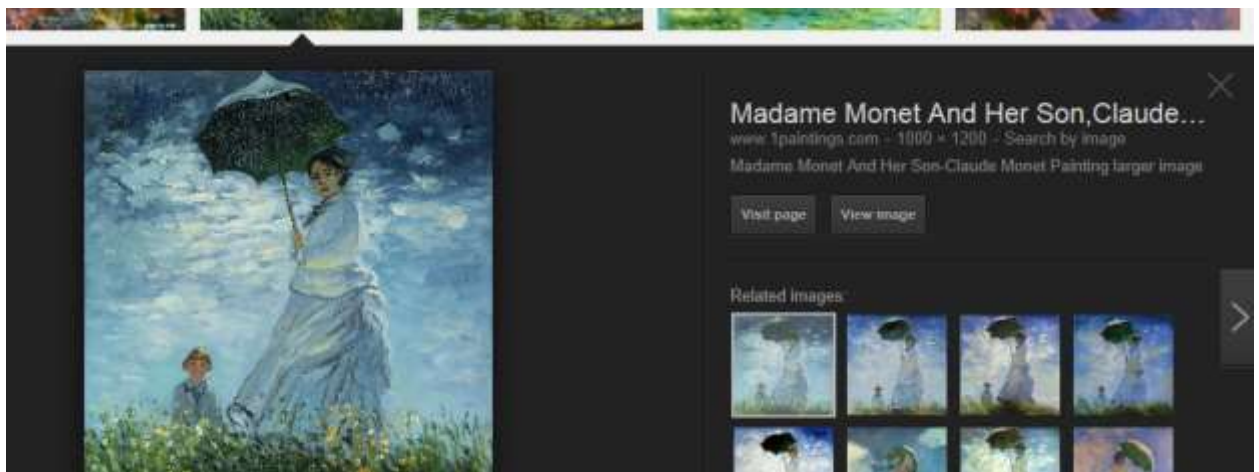
type of image: Limit the kind of images you find.

region: Find images published in a particular region.

site or domain: Search one site (like .edu, .org or .gov) or limit to a specific site.

Then go to the bottom of the page and click “advanced search”.

- Google Images will then show thumbnails of images that meet the search criteria that we defined. To collect the digital images and save them to the hard disk, you should click the thumbnail of the image that you want to save. Google Images will open a small window with the image displayed in the left side of it.



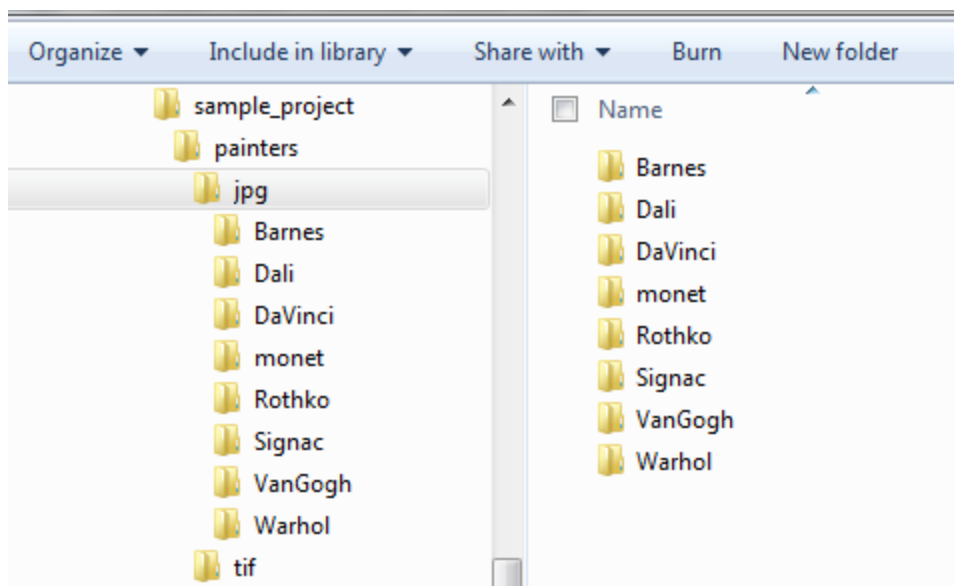
In that window, right click the larger image in the left side of the window, and then select “save picture as...”, and save the image to the folder of Claude Monet paintings. That ensures that the file saved to your hard drive is the large image file, and not the small thumbnail, which is useless for computer analysis. The collection of digital paintings should be repeated until at least 30 digital paintings for each artist are obtained.

Obviously, you should avoid using images that have watermarks, labels, frames or any other feature that is not in the original painting. Such content that was not created by the artist can add noise to the data and degrade the ability of the computer to analyze the artistic style. Here are some examples of images that should **not** be part of your dataset.



7. **Organizing the image files in folders:** The organization of the image files in folders is very important for the experiment, and the processing of the files depends on the correct organization of the files in folders. Saving the files in a fashion that does not follow the correct folder structure might require substantial efforts in organizing the files after they are downloaded and stored.

The image files should be saved to the hard drive such that each group of image files used in the experiment is saved in a different folder. For instance, in experiment 1.1, each group of image files is a painter. Therefore, the experiment folder will have sub-folders of different painters:

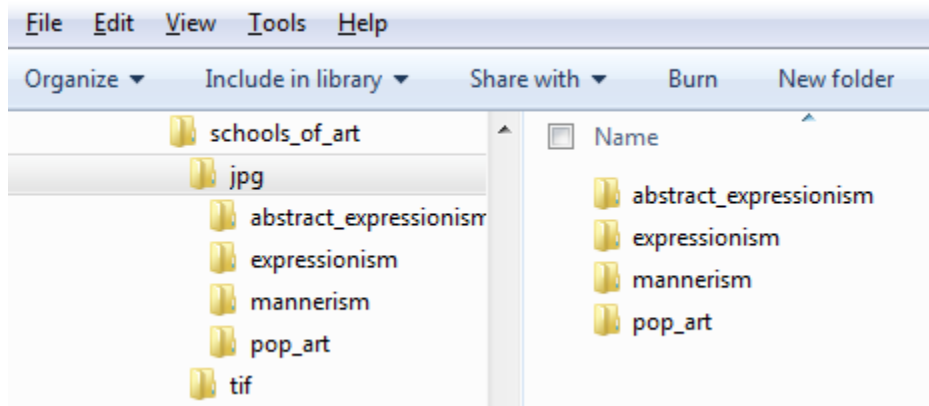


Each painter has a sub folder in which the image files should be stored. For instance, all image files of the paintings of Mark Rothko should be placed in the folder “Rothko”, while all image files of the paintings of Paul Signac should be saved to the folder “Signac”. It is important to **avoid using space**

in folder names, as spaces might make the command line more complicated after we copy the folders for processing in the cluster.

All painter folders in the example are sub folders under the folder “jpg”. The reason is that these are the JPG files downloaded from the internet. The processing will be done after converting the JPG files to TIF files, as will be explained later in this document.

In the case of an experiment that is based on schools of art, each image file of a painting needs to be saved in the folder of that school of art. For instance, all images of mannerism paintings should be saved in the “mannerism” folder.



3. Preparing the image files for processing (pre-processing)

The image files collected from the internet need to be processed before they can be analyzed by the computer. The processing includes two primary tasks: converting the JPG files to TIF files, and normalizing the size of the files. Fortunately, these two tasks can be achieved in one step using a simple image processing tool.

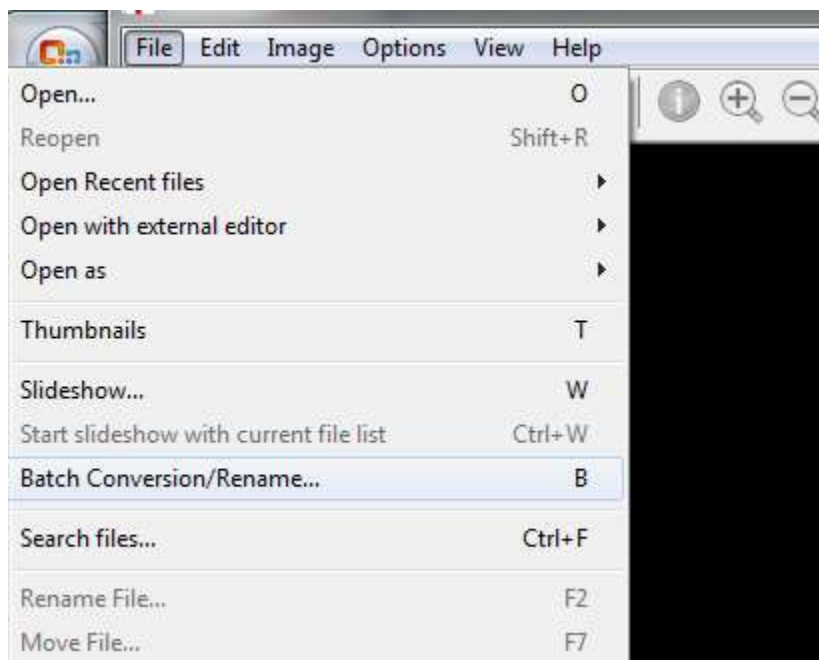
Converting JPG to TIF: The JPG (Joint Photographic Expert Group) image format is a “lossy” compressed file format. That is, the files are designed to store image information such that the size of the file is as small as possible, and therefore browsers can quickly download and display the images. The image analysis tool that will be used later in this document reads TIF images, and therefore we need to convert all JPG files to TIF files. Many image analysis tools and on-line services can perform such file conversion. In our experiment we will use a software called “IrfanView”, which can convert the file format and normalize the image sizes at the same time.

Normalizing the image size: The image files collected from the internet are all larger than 800x600 pixels, but are still of different sizes. For instance, a file of 1500x1200 pixels is also larger than 800x600, and therefore can be included in our dataset. However, analyzing files of different sizes might add noise to the analysis done by the computer, since the size of each image file can affect the way the image file

is analyzed. Therefore, we need to *normalize* the image files so that each image file has the same number of pixels.

To convert the files to the TIF file format and to normalize the number of pixels, we can use a simple yet effective image processing tool called IrfanView. To use the tool, we will first need to download and install it. IrfanView can be freely downloaded from its web site <http://www.irfanview.com>. Then, click on the “download” link and select the source from which you want to download the program.

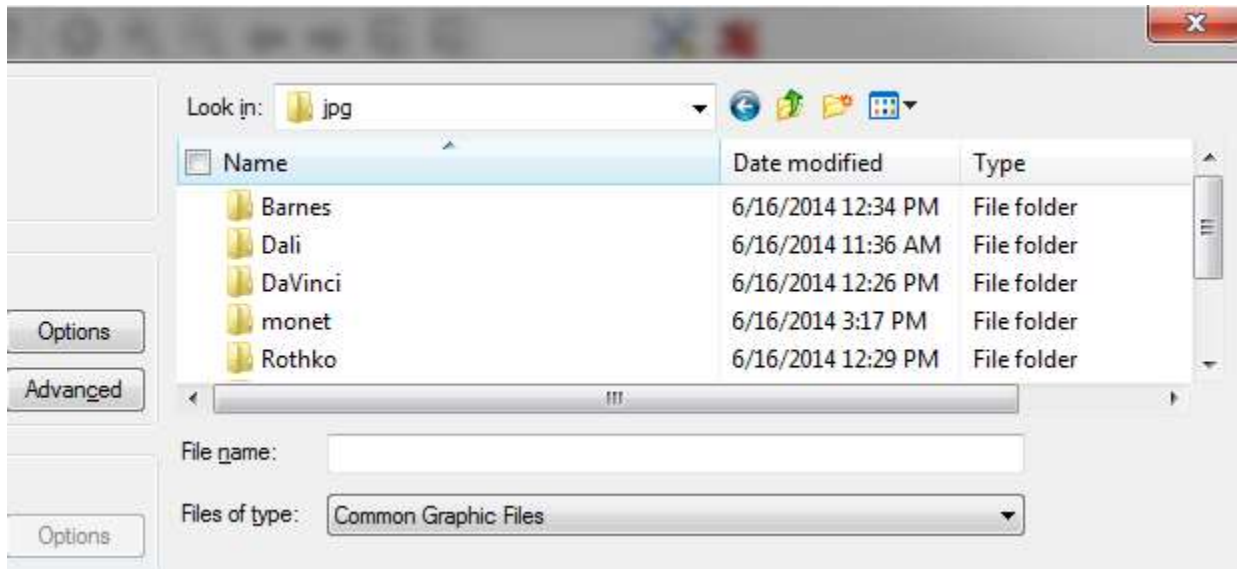
After IrfanView is installed on the computer, we can use it for file format conversion and image file normalization. For that, we first need to run IrfanView, click on the “File” menu at the top left of the window, and then select “Batch Conversion/Rename...”.



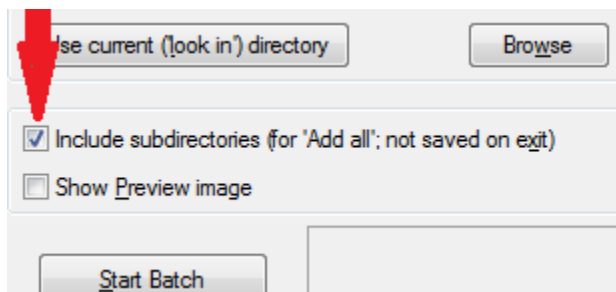
What is batch processing? Batch processing is the execution of a certain command or sequence of commands multiple times without human intervention. When applying the same task to multiple files, batch processing can very quickly complete the processing of all files with very little efforts from the user. Since we apply the same commands (size normalization and format conversion) to all of our image files, we can use batch processing to avoid repeating the same tasks for each file separately.

The “Batch conversion” window will open. To normalize and convert the file format of the image files in one batch you should do the following:

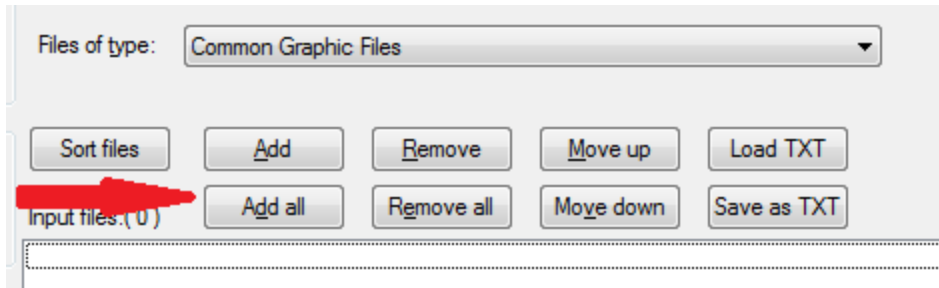
1. In the top right part of the batch conversion window there is a file selection box titled “Look in:”. You should click that box and select the root folder of the subfolders of the image files you downloaded from the internet. That will tell IrfanView to search for all files in that folder. After selecting the root folder, you should see the sub folders in the small window below the “Look in:” box.



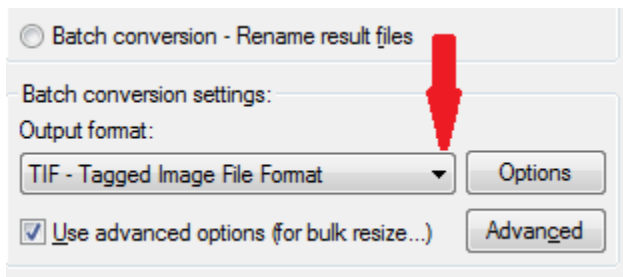
2. Now we need to tell IrfanView to look for image files inside these sub-folders. For that, we need to check the checkbox “Include subdirectories” at the bottom left of the batch conversion window (above the “Start Batch” button).



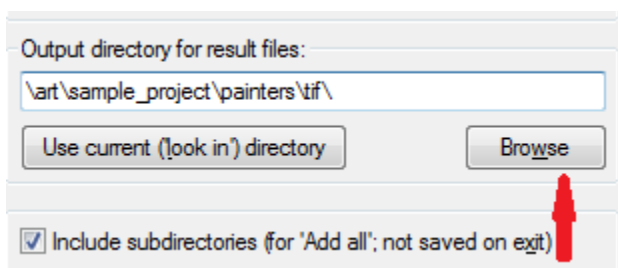
3. Now, you can click the “Add all” button. All image files in the subfolders will be shown in the window below that button. These are the image files the IrfanView will process and convert.



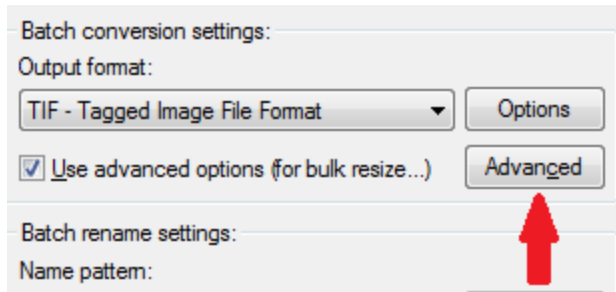
4. Now we need to select the output format. As mentioned above, we want to convert all JPG files to TIF files. For that, we need to select “TIF – Tagged Image File Format” in the “output format” drop down list in the top left part of the screen. Also, make sure that the checkbox “Use advanced option (for bulk resize...)” is also checked.



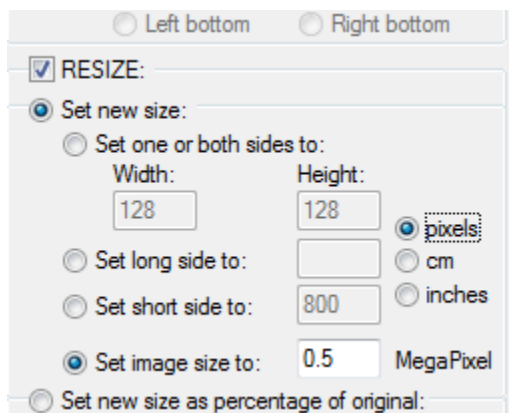
5. We also need to specify the output folder in which IrfanView will save the converted files. That is done by clicking the “Browse” button of the “Output directory for result files:”, and then selecting an output folder, which has to be different from the input folder.



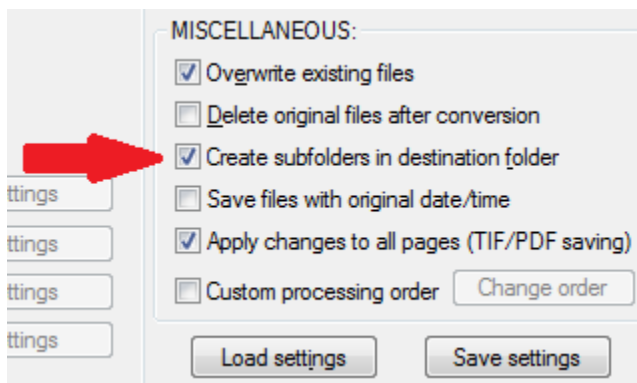
6. Steps 1-5 above complete all necessary steps to convert all files from JPG to TIF. However, we also want to normalize the size of the image files. For that, we need to click the “Advanced” button to open the advanced image processing option window.



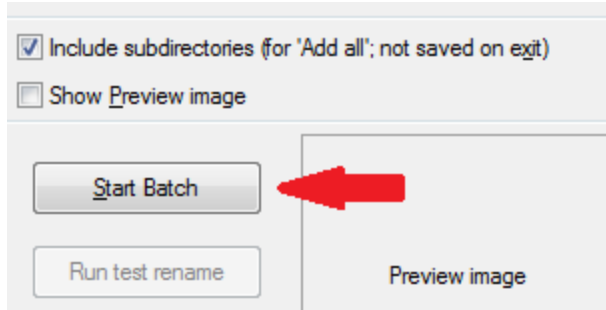
- In the advanced image processing window, check the “RESIZE” checkbox, and then click the “Set new size:” button. Then, select the “Set image size to:” button and type “0.5” in the edit box. That will tell IrfanView to set all image files to a new size of 500,000 pixels.



- In the “MISCELLANEOUS:” section, make sure that the “Create subfolders in destination folder” button is checked.



- When done, click the “OK” button at the bottom right of the window. That will bring back the “Batch conversion” window. Now we just need to click the “Start Batch” button in the bottom left corner of the screen. That will start the batch conversion and processing of the images.



The batch conversion will take a few seconds, depends on the number of images you need to process. In the end of the batch conversion, all files will be in the output folder that you specified. The image files will be in subfolders created automatically by IrfanView, and all files will be of the same number of pixels, and in the TIF file format. The output of IrfanView will be the input of the image analysis software that will be described in the next section.

4. Copying files to the server

Each image file is a large collection of pixels, that together contain all information about the visual content. While these pixels can be displayed as an image that can be easily perceived by the human brain, machines are much less effective in working with large collections of pixels. To allow machines to work effectively with images, we first need to represent each image by a set of numerical value that reflect the image content. These values are called “numerical image content descriptors”, or “image features”.

What is an image feature? An image feature (or “numerical image content descriptor”) is a number that reflects the visual content. For instance, the number of red pixels in the image is a value that reflects the visual content, in that case how red the image is. That is obviously a very simple numerical image content descriptors. Many other content descriptors can be much more complex, and can reflect very many aspects of the visual content such as shape, color, textures, luminosity, fractals, and more. In our experiment we use the *Wndchrm* tool to compute a large set of numerical image content descriptors. Each feature contains a small amount of information about the painting, but all features together contain substantial amount of information that reflects many different aspects of the visual content that we process.

In the experiment we will use an image analysis tool called *Wndchrm* (Shamir et al., 2008), which is an open source and freely available software. *Wndchrm* requires substantial computing resources, which are not normally available when using standard laptops or desktop computers. If you work on a powerful PC you might be able to compute the numerical image content descriptors on your own computer, and that process might take several hours or even a few days to complete. So if you work on

a powerful computer that has four or more cores and you have no more than around 150 images you can just compute all files on your working computer without transferring the files to the server. That, however, might easily take a day or sometimes more to complete. To reduce the response time, the processing of the images will be done on a server with multiple processors.

What is Wndchrm? *Wndchrm* is a command-line program that can analyze complex images automatically. It computes a large set of ~4000 numerical image content descriptors from each image. These numbers include a comprehensive set of numerical values that reflect the visual content such as textures, shapes, colors, edges, fractals, statistical distribution of the pixel intensities, coefficients of the polynomial decomposition, and more. These features are also extracted from transforms of the image such as the Fourier (frequency domain) transform, Chebyshev (polynomial decomposition) transform, and wavelet transform. It can perform classification of images, and analyze the similarities between different groups of images.

Wndchrm was originally developed to analyze microscopy images, but its ability to analyze complex morphology made it useful also for analyzing visual art. More information about *Wndchrm* can be found in its download page at <http://vfacstaff.ltu.edu/lshamir/downloads/ImageClassifier>.

Description of *Wndchrm* can be found at [Shamir, L., et al., Wndchrm - an open source utility for biological image analysis, BMC Source Code for Biology and Medicine, 3: 13, 2008.](#)

To use the server, we first need to transfer the files from our laptop to the server. That will be done by using the **PSCP** utility. *Pscp.exe* is a compact computer program that can be used to transfer files between computers over an internet connection. It is the Windows version of the commonly used Linux SCP (Secure Copy) program. With *PSCP* you can transfer files from your laptop to any other server on which you have an account, or copy files from any server to your laptop. *PSCP* is a free software that can be downloaded from <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>

Detailed description and instructions about *PSCP* are available at <http://the.earth.li/~sgtatham/putty/0.60/html/doc/Chapter5.html>.

What is secure copy? *SCP* (Secure Copy Protocol) or its Windows equivalent *PSCP* is a software that implements the Secure Copy Protocol algorithm, which allows transferring data over the network while ensuring the authenticity and confidentiality of the data being transferred. It allows copying a file from one computer to a remote computer through the internet. Before transferring the file over the network, the file content is encrypted using a public key, and then decrypted on the remote computer using the public key and the private key of the remote computer. With *SCP* other computers sharing the same network but do not have the private key cannot know the content of the file.



To download *pscp.exe*, you need to open the download page (<http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>), and then look under “Binaries” for *pscp*. We will also download *putty.exe*, which will be used later to login to the server after the files

are copied. To download these two files, we need to right click on the file name and select “Save target as” or “Save link as”, depends on your browser.

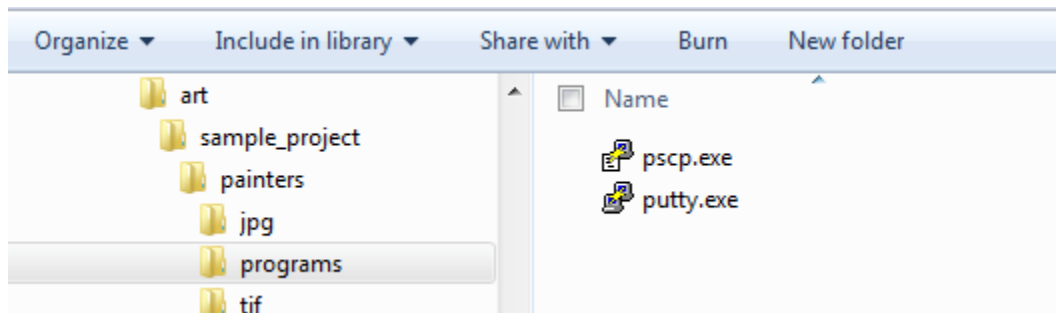
Binaries

The latest release version (beta 0.63). This will generally be a version I think is reasonably likely to work worth trying out the latest development snapshot (below) to see if I’ve already fixed the bug, before report

For Windows on Intel x86


PuTTY:	putty.exe		(or by FTP)	(RSA sig)	(DSA sig)
PuTTYtel:	puttytel.exe		(or by FTP)	(RSA sig)	(DSA sig)
PSCP:	pscp.exe		(or by FTP)	(RSA sig)	(DSA sig)
PSFTP:	psftp.exe		(or by FTP)	(RSA sig)	(DSA sig)
Plink:	plink.exe		(or by FTP)	(RSA sig)	(DSA sig)
Pageant:	pageant.exe		(or by FTP)	(RSA sig)	(DSA sig)
PuTTYgen:	puttygen.exe		(or by FTP)	(RSA sig)	(DSA sig)
A .ZIP file containing all the binaries (except PuTTYtel), and also the help files					
Zip file:	putty.zip		(or by FTP)	(RSA sig)	(DSA sig)

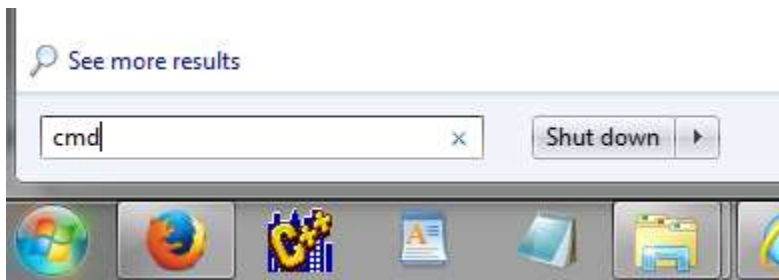
Then you need to select the folder in which the files will be saved. It is convenient to create another folder named “programs”, in which all programs used the experiment will be saved. Then we can save the pscp.exe and putty.exe in that folder.



Pscp is a command-line utility, which means that it can only run through the terminal (command-line) window. Trying to double-click pscp.exe like any other program we wish to run will open the terminal window for a fraction of a second, and then the program will close without doing anything useful.

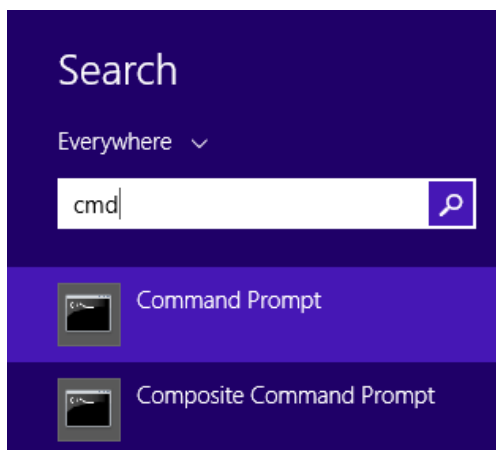
What is a command-line interface? Unlike graphical user interface in which the user controls the computer using a mouse or touchscreen, in a command-line interface all interactions with the computer are done through text commands. The user types commands as input, and the computer produces the output on the same window. Command-line interfaces were very common before Windows became popular, and are still used, normally in servers. Most operating systems such as Mac OS and Windows also have the option to control the computer through a command-line interface. Command-line interfaces require to memorize the commands, but on the other hand they allow running complex commands performing tasks that cannot be done by using the graphical user interface.

To use *pscp*, or any other command-line tool, we first need to open the terminal window. In Windows 7, that can be done by clicking the  button at the left of your taskbar or pressing the right Windows key on your keyboard, and then typing “cmd” in the search box.



Then select “cmd.exe” to open the command line window.

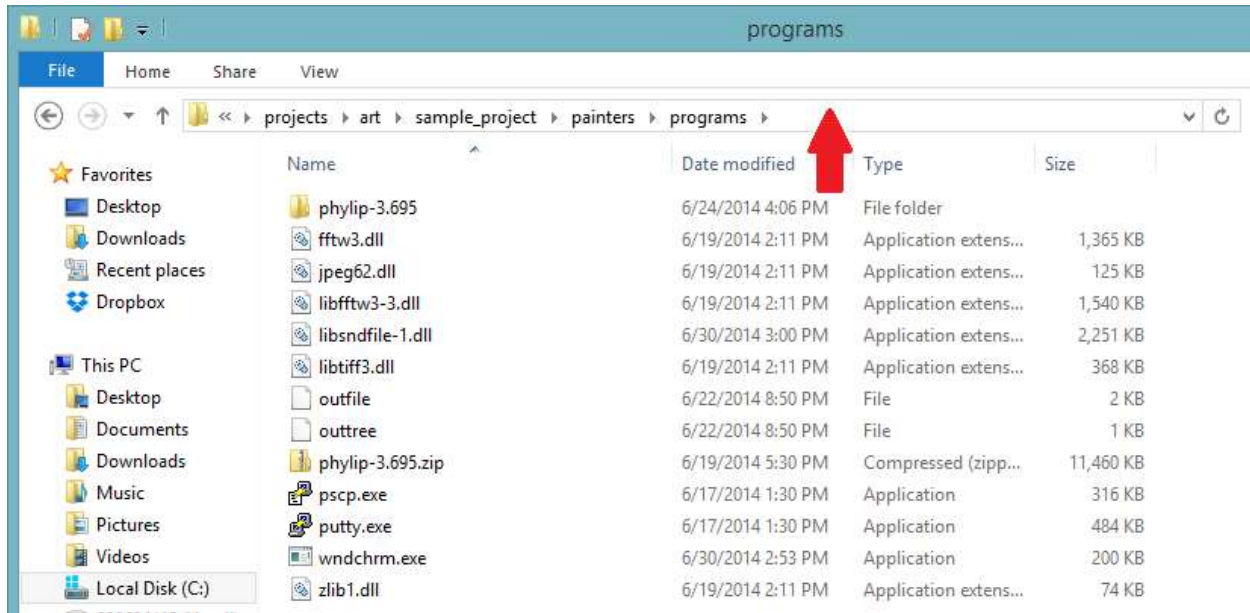
In Windows 8.1, you need to click the Windows  button and then type “cmd” or “command prompt”.



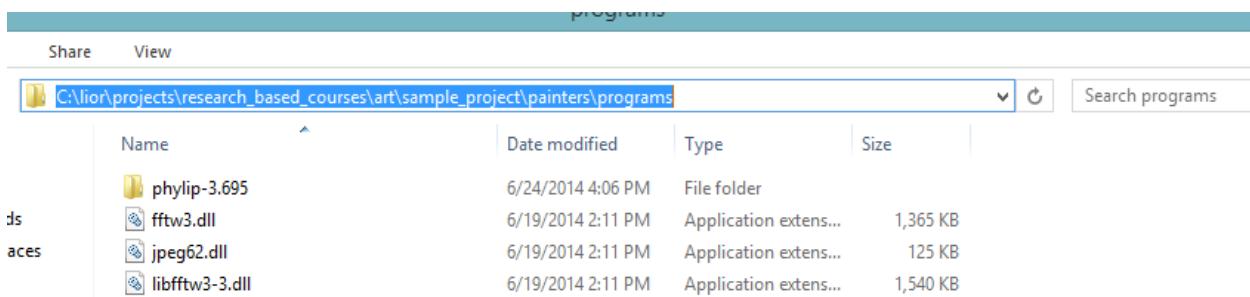
Then, select “Command Prompt” to open the command-line interface window.

To use the program pscp.exe you first need to change the directory to the folder where pscp.exe was saved. For that you will need to use the command `cd` (Change Directory). In your command prompt window, type “`cd`”, followed by the full path to the folder where your pscp.exe file is.

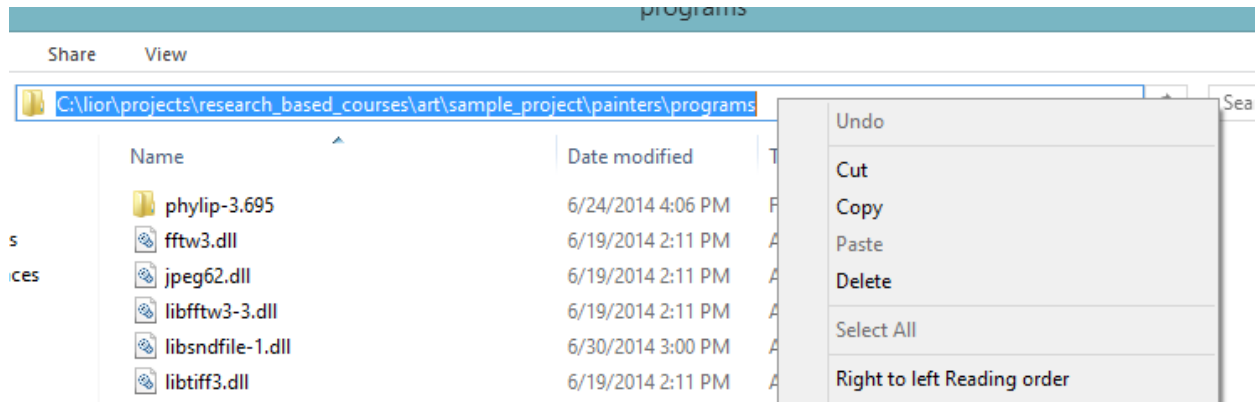
If you do not know the full path to the folder where pscp.exe is, you can find it with your file explorer, and then click the box near the top of the file explorer window, right of the name of the folder (“programs” in this case), as marked with the red arrow below:



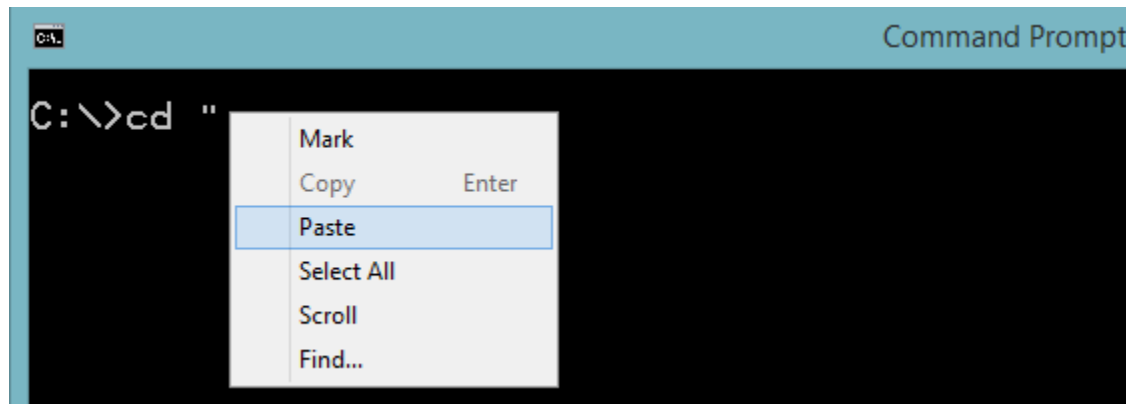
The box will then show the path of the folder, and the text will be highlighted.



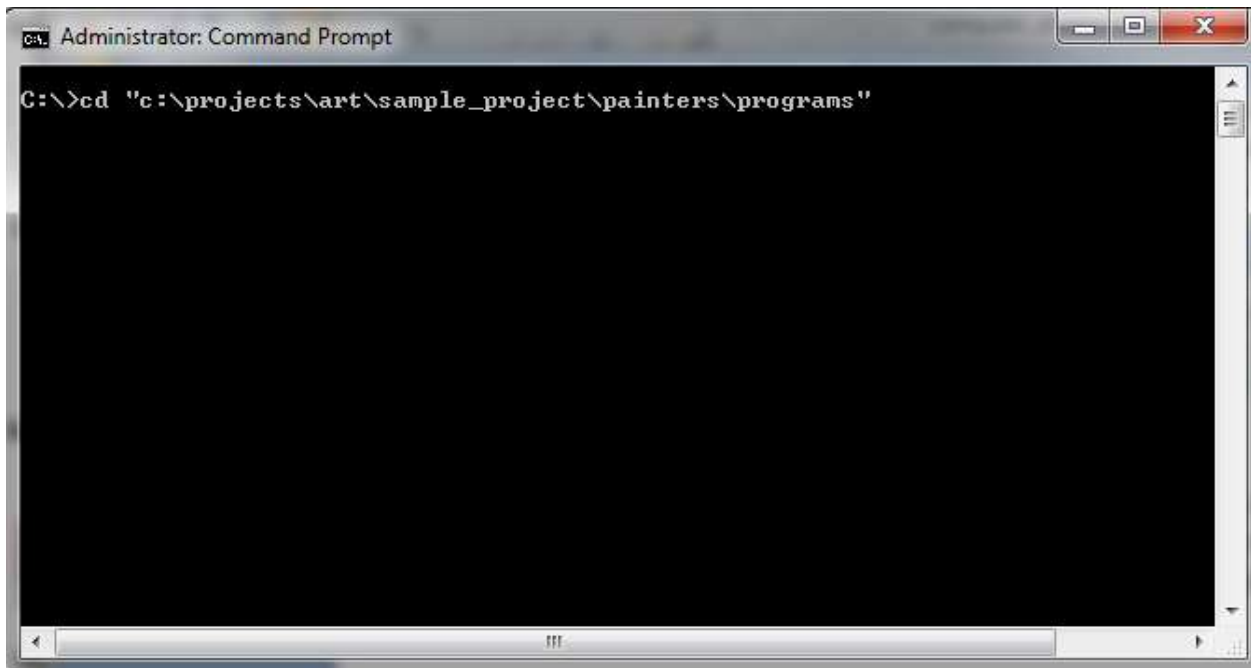
Then you can right-click the text box and select “copy” to copy the path to your clipboard.



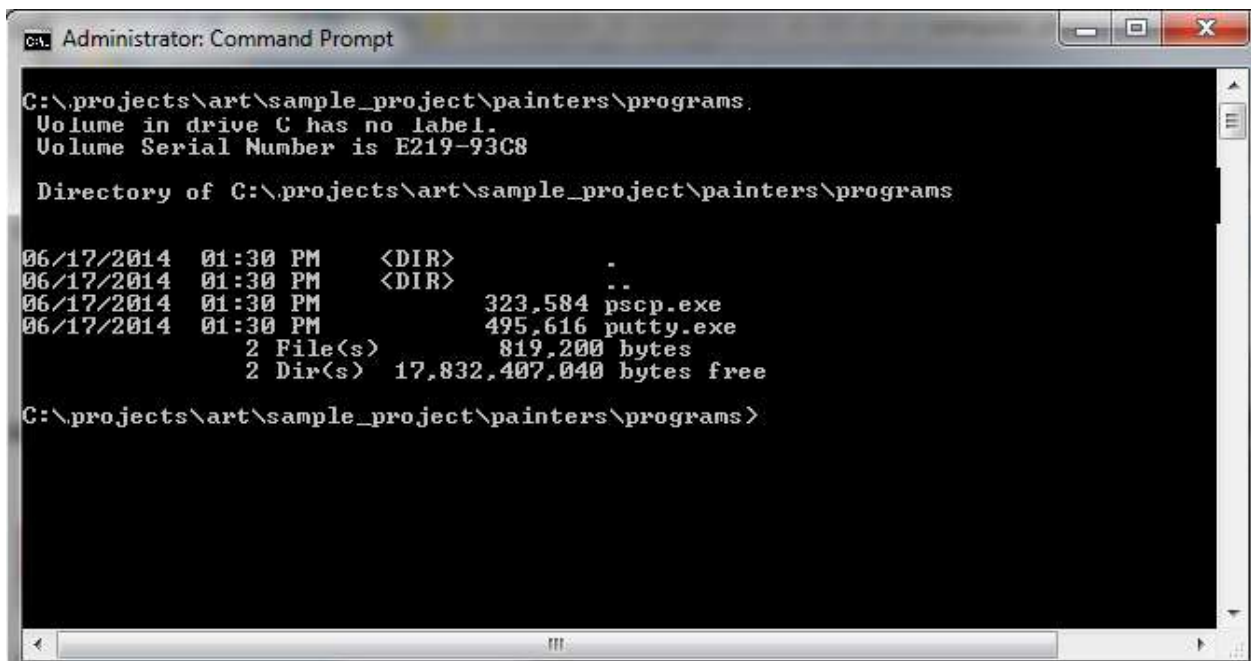
And then paste it to the command line window by right-clicking the command prompt window and selecting “paste”. The path should be inside quotation marks (“”).



Then press <Enter>. If you copied the path manually, make sure to type the full path correctly.



After you press <Enter>, the command prompt will show the path. Typing the command "*dir*" will show a list of tiles, which is the content of the directory. In our case the content of the folder is just the two files that we saved in it: *pscp.exe* and *putty.exe*.



If your default drive is not the C drive, you will see a different letters at the left of the path. For instance, the path will start with "M:\>" if your default drive is M. In that case, you will simply need to type "C:" and press <enter> to change the current drive to C.

Now, if we type “*pscp*” we will see a short description of how to use *pscp*. That is because *pscp.exe* requires some arguments, and if the arguments are not provided it just displays the instructions. To use *pscp*, you will need to know the username and password of the account that was created for you on the server.

The synopsis of the command line for copying your files is:

```
pscp -r PathToRootFolder username@servername:
```

Pscp is the name of the program that copies the files to the server, “-r” is a switch that tells *pscp* to copy the entire folder, including the sub folders, *PathToRootFolder* is the full path to the root folder in which our image files are stored, in the example above it is “c:\projects\art\sample_project\painters\tif” (the folder in which the processed TIF files are stored). *Username* is the user name assigned to you when the account on the server was created, and *servername* is the name or IP address of the server that we want to copy the files to.

For instance, assuming that your username is “group1” and the IP address of the server is 207.73.64.26, the command line will be:

```
pscp -r “c:\projects\art\sample_project\painters\tif” group1@207.73.64.26:
```

```
C:\projects\art\sample_project\painters\programs> pscp -r “c:\projects\art\sample_project\painters\tif” group1@207.73.64.26:
```

Note the colon (:) in the end of the command.

That command tells *pscp* to copy all files and subfolders inside the folder “c:\projects\art\sample_project\painters\tif” to the server. It will create a folder named “tif” in your account on the server, and all files and subfolder will be copied to it.

After pressing <Enter>, *pscp* will ask for your password. Note that when typing the password you will not see the keys that you typed, so you just need to assume that you typed the password correctly.

```
C:\art\sample_project\painters\programs>pscp -r \art\sample_project\painters\tif
lshamir@skynet.mtsu.edu:
Using keyboard-interactive authentication.
Password: _
```

You should type your password and press “Enter”, and then the files will be copied to the server. *Pscp* will display on the screen the files that are being copied.

```
C:\art\sample_project\painters\programs>pscp -r \art\sample_project\painters\jpg
lshamir@172.18.16.106:

Using keyboard-interactive authentication.
Password:
Monet_-_H_Euser_in_Argente : 72 kB | 72.8 kB/s | ETA: 00:00:00 | 100%
Water-Lilies-and-Japanese : 380 kB | 380.8 kB/s | ETA: 00:00:00 | 100%
C:\art\sample_project\painters\programs>
```



After copying the “tif” folder, you can also copy the jpg folder to your account on the server by using:

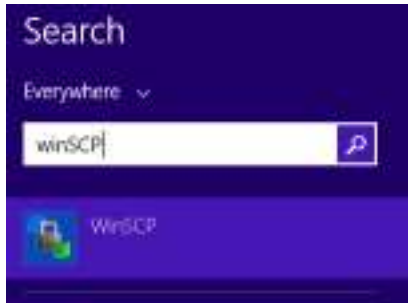
```
pscp -r "c:\projects\art\sample_project\painters\jpg" group1@207.73.64.26:
```

```
C:\projects\art\sample_project\painters\programs> pscp -r "c:\projects\art\sample_project\painters\jpg" group1@207.73.64.26:
```

What is an IP address? IP (Internet Protocol) address is a number that identifies a certain computer connected to the internet. No computer can be accessed via the internet unless it has a unique IP address. An IP address is a number made of four numbers in the range [0..255], separated by a dot. The domain names that we commonly use to access web sites (e.g., www.google.com) are associated with IP addresses using a Dynamic Name Server (DNS). The computer first resolves the IP address from the domain name, and then it can access the data in that web site. For instance, you can find the IP addresses of any web site (e.g., www.google.com) by typing into the command prompt “ping www.google.com”.

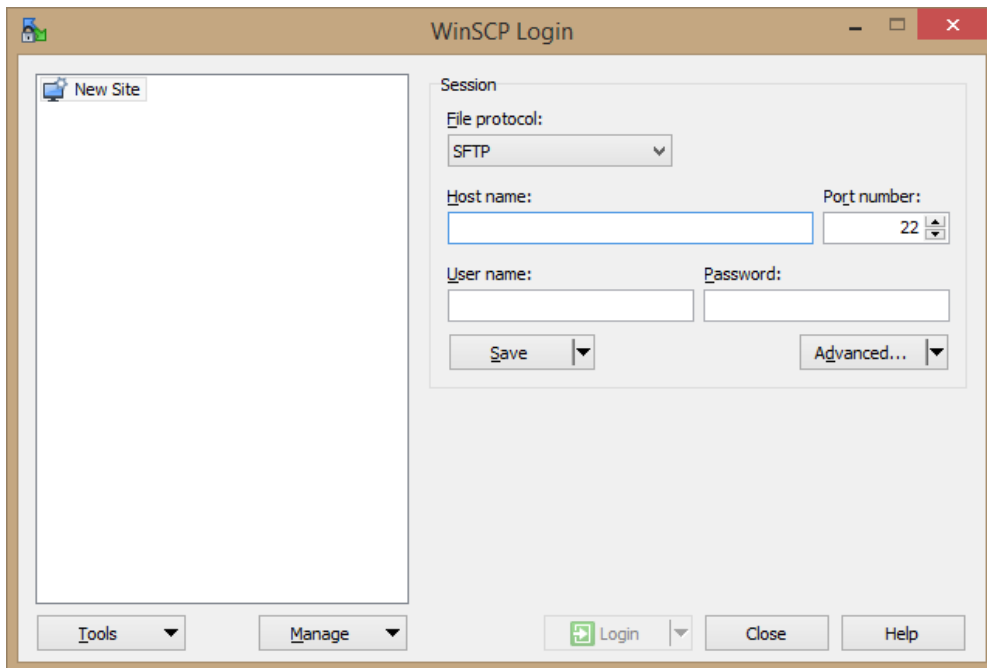
5. Viewing files on the server

After the files are copied to the server, you can view the files using the program WinSCP. WinSCP is a secure file transfer utility that runs in Windows with a graphical user interface. It can be used to view and copy files from a local computer to a remote server. To start WinSCP, you need to click the  button in Windows 8.1, or the start button  in Windows 7, and type “WinSCP” followed by <Enter>.



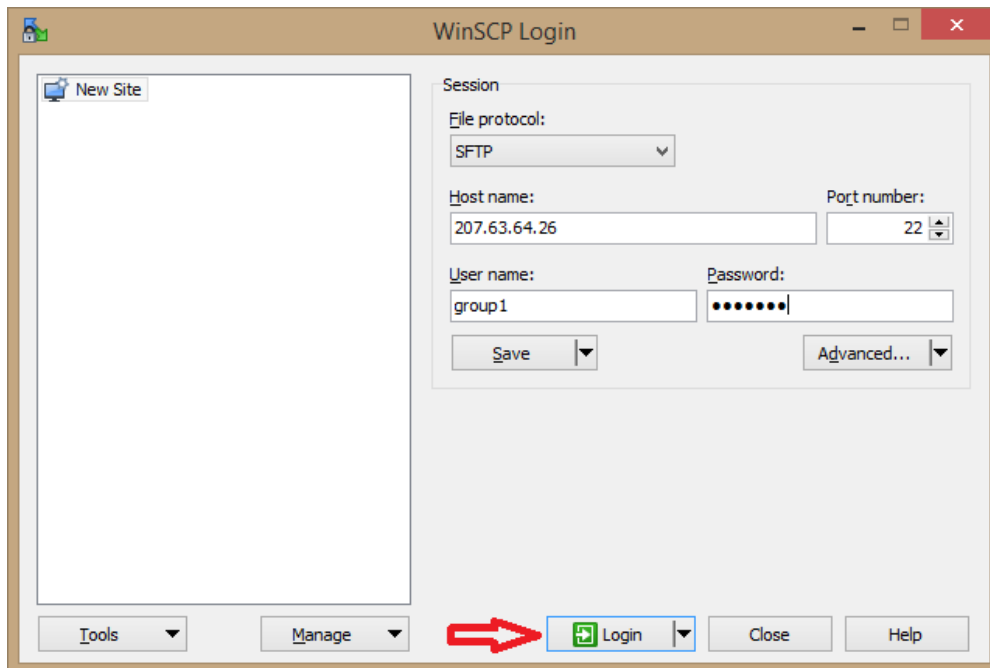
If WinSCP is not installed on your computer, you can download and install it from <http://winscp.net/eng/download.php>

After launching WinSCP, you will see a dialog box asking for the server information.



The host name is your remote server name or IP address. For instance, 207.73.64.26.

The username and password are the same username and password you received. Then you need to click the "Login" button.



After logging in, you will see a folder structure of your files. You will also be able to copy files to and from the server. Just like pscp, WinSCP can be used to copy your files to the server, and then download them back to your computer.

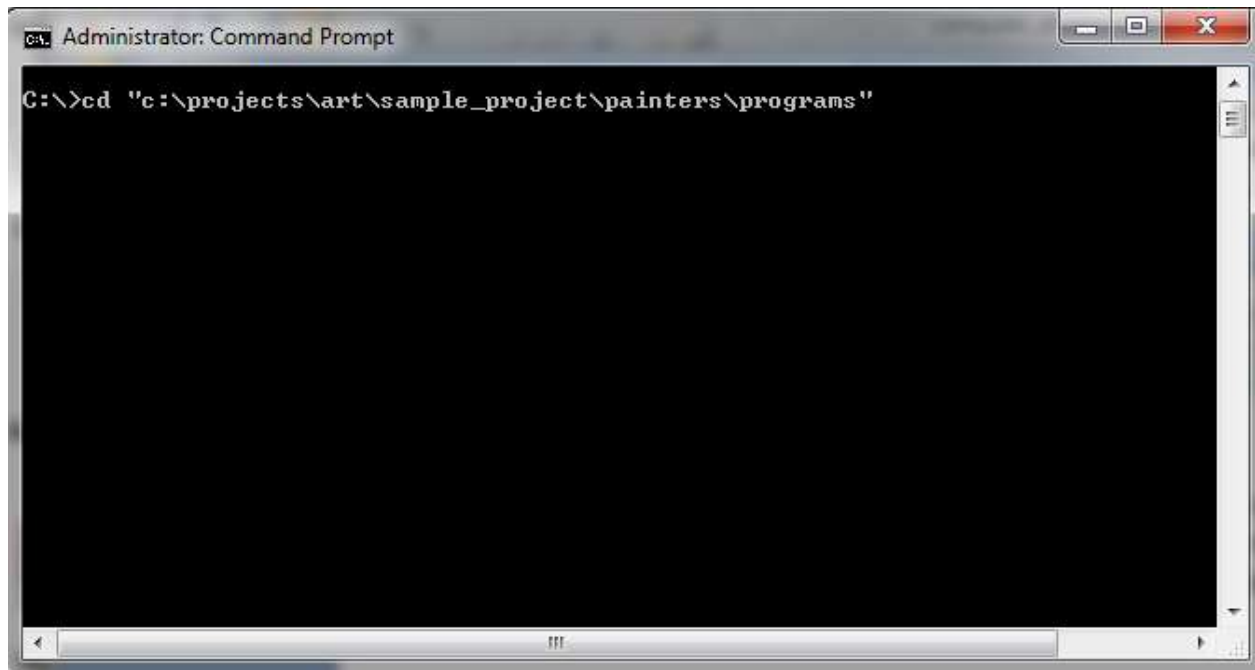
6. Processing the files on the server

After the files are copied to your account on the server, *Wndchrm* can be used to process these files and compute the numerical image content descriptors. For that, we will use the *putty* program that we downloaded in the previous step.

What is Putty? Putty is the Windows version of the program SSH (Secure Shell) used in Unix, Linux, and Mac OS. The program opens a command-line window that allows running commands on a remote computer. That is, the command-line window opens on one computer, but the commands written in it are being executed on another computer. The computer receives the IP address of the computer it aims at controlling, as well as the name of the user. Obviously, the user must have an account on the remote computer to allow access. All commands are encrypted on the local computer before being sent to the remote computer, and the output being sent back from the remote computer is also encrypted so that all communications are confidential to all other computers connected to the same network.

Putty is a program that allows to connect and work on other computers through the network. It opens a terminal window, which is actually another computer. Obviously, to connect to another computer you need to have a valid account on that computer, which will allow you access.

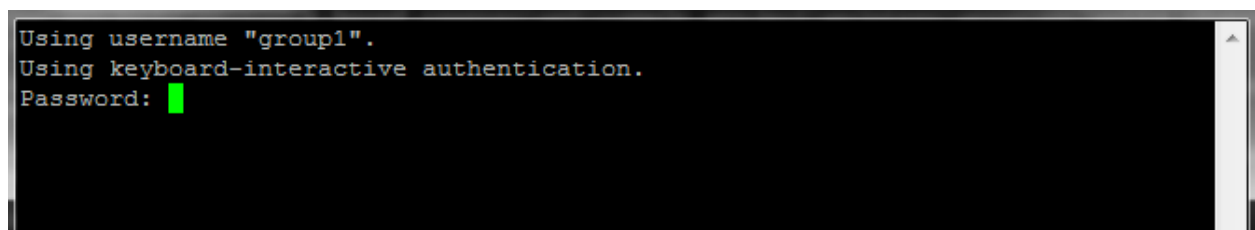
First, open your command prompt window as you did for using *pscp*. Again, you will need to change the current directory using the “cd” (Change Directory) command. You will need to set the working directory to the “programs” directory, in which you saved *putty* when you downloaded it.



Now you can run *putty*. For instance, if your username is “group1”, and the IP address of the server is “207.73.64.26”, the command line is “putty -l group1 207.73.64.26”, as the following:

```
C:\projects\art\sample_project\painters\programs>putty -l group1 207.73.64.26
```

Putty will open the terminal window and will ask for your password.



After typing the password and pressing <Enter>, *putty* will log you in. After logging in, through the terminal window you will work on the server, not on your own computer.

Now we can run *Wndchrm* to process the images. That can be done by using the following command:

```
wndchrm train -mlc -t4 /home/group1/tif /home/group1/tif/output.fit &
```

The “-m” switch tells *Wndchrm* to save the numerical content descriptors in files, so we can then copy these files to the laptops. The “-l” switch tells *Wndchrm* to use the larger feature set, and the “-c” switch tells *Wndchrm* to compute color features. The “-t4” tells *Wndchrm* to separate each image to 4x4 equal-sized tiles, and process each tile of the image separately.

To complete the process faster, you can also run more than one instance of *Wndchrm*. For that, you just need to retype the command again. Each command will run another instance of *Wndchrm*, but you should not run more than 10 instances.

When *Wndchrm* starts to run, you will see the names of the files that are being processed displayed on the screen.

Processing all files will take several hours or even several days, depends on the number of images. However, the server will work without further attention. If you are interested to know if the server has processed all files, you can check if the file `/home/group1/tif/output.fit` exists. To do that, you can login to the server as described above and then type:

```
ls /home/group1/tif/output.fit
```

Assuming that your username is “group1” (if the group name is different the command should be changed accordingly). If the file is not found, *Wndchrm* has not done processing the files, and will show a “No such file” message such as:

```
@localhost home]$  
@localhost home]$  
@localhost home]$ ls /home/group1/tif/output.fit  
ot access /home/lshamir/tif/output.fit: No such file or directory  
@localhost home]$
```

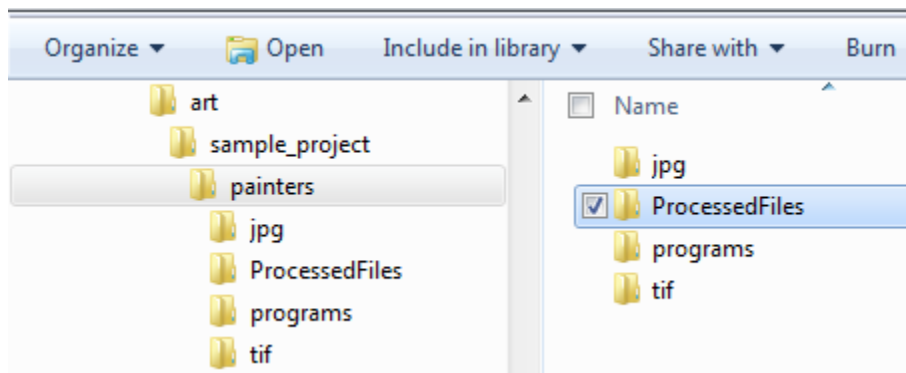
When you are done, you can simply type “logout” followed by <Enter> to end your session with the server.

```
@localhost home]$
@localhost home]$
@localhost home]$ logout
```

7. Copying the files from the server to the laptop

After the files are processed, you will need to copy all the processed files back to your computer for further analysis. That is done again with pscp.exe, but this time instead of copying files from your computer to the server, we will copy the files from the server to the laptop.

To do that, first we need to create a folder to which we can download all the processed files from the server. For instance, under the folder “painters” we can create another folder called “ProcessedFiles”.



Then, we again need to open the command prompt window and change the directory to our directory where pscp.exe is located (using the command “cd”). For instance:

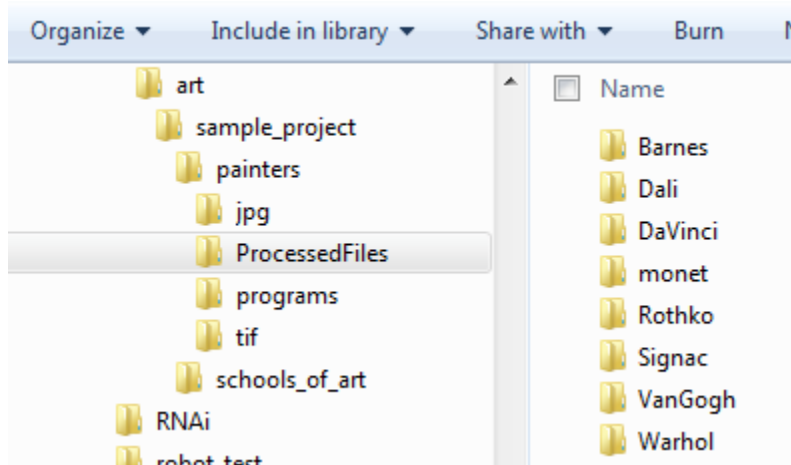
```
cd “c:\projects\art\sample_project\painters\programs”
```

Then, we need to use pscp.exe in the following way:

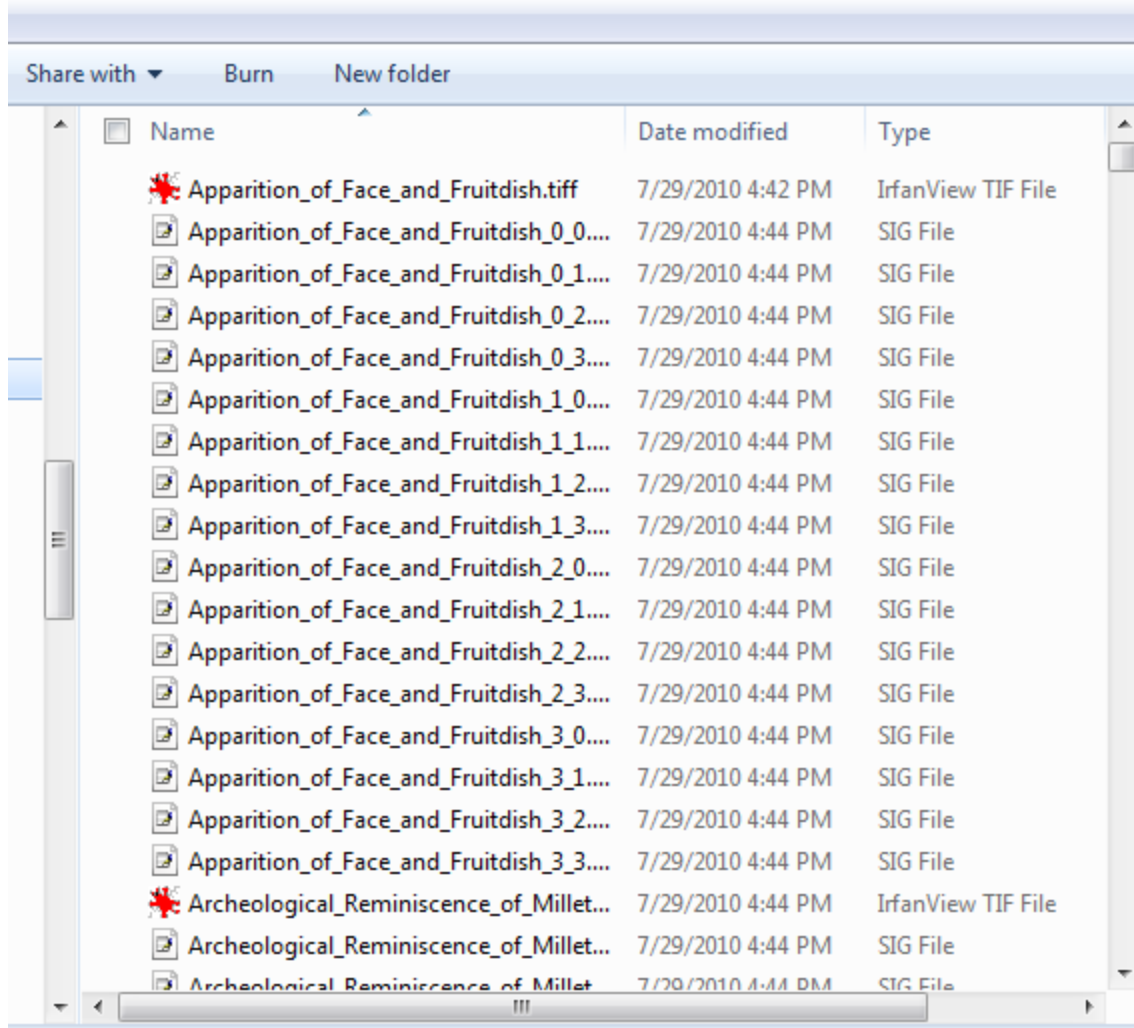
```
Pscp.exe -r group1@207.73.64.26:tif “c:\projects\art\sample_project\painters\ProcessedFiles”
```

The username “group1” should be changed based on your user name that you were given, and the target path (in the example “c:\projects\art\sample_project\painters\ProcessedFiles” should be changed to where your project is on your hard drive. That command will tell pscp to copy all files and subfolders inside the folder “tif” on the server to the folder “c:\projects\art\sample_project\painters\ProcessedFiles” in your computer.

Pscp will ask for your password, and after you type your password and press “Enter” all files will be copied to the folder “ProcessedFiles”.



If you look at the content of the subfolders you will notice that each TIF image file has 16 files with similar names, but the extension of these files is “.sig”.



The .sig files are the files created by Wndchrm when the files were processed by the server. Each .sig file contains the numbers that describe the visual content of the paintings (the image numerical content descriptors). Each image file has 16 .sig files because each tile is computed separately, so there is one .sig file for each of the 16 tiles of the image.

8. Analyzing the files and creating phylogenies

To analyze the files, we will use the Wndchrm program. Because the image files are already processed, Wndchrm will not process the image files again, but instead will read the values from the .sig files. That allows performing the image analysis on a laptop.

To perform the analysis on the laptop, we will need to download the Wndchrm to our computers. Wndchrm can run on Linux servers, but it also has a version that can run on Windows. The files can be downloaded from <http://people.cs.ksu.edu/~lshamir/downloads/ImageClassifier/>

In the download web page, you should look for the Windows files.

MS-Windows users can download a binary executable file [wndchrm.exe](#)

The executable file needs the following run-time DLLs:

[fftw3.dll](#)

[libfftw3-3.dll](#)

[libtiff3.dll](#)

[jpeg62.dll](#)

[zlib1.dll](#)

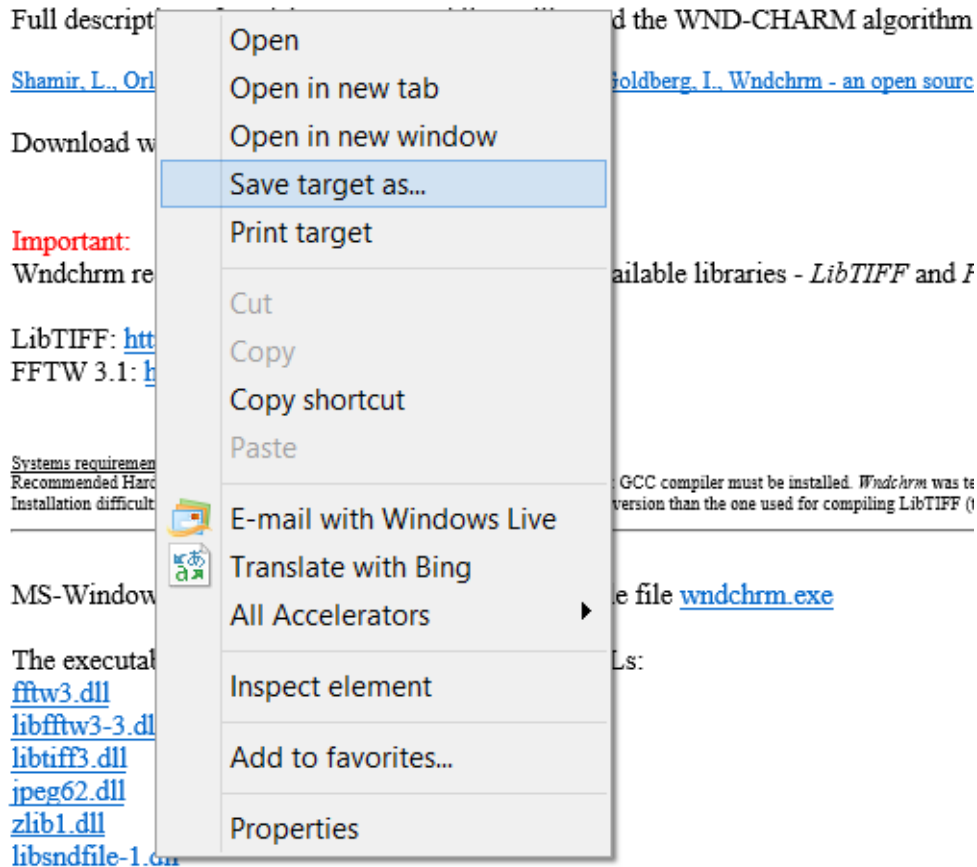
[libsndfile-1.dll](#)

MS-Windows developers can download compiled libraries for [Borland C++](#) and [Microsoft Visual C++](#)

You will need to right click the file “wndchrm.exe” and choose “Save target as...” or “Save link as...” and save the file in your “programs” folder.

WND-CHARM - A multi-purpose image classifier.

The `wndchrm` command-line utility gets a root directory as a parameter and compute algorithm. It can then split the images into training and test sets and classify them. Tl for instructions.



When you download the file, your browser might not recognize the file and might therefore give you a warning.



In that case, you should click “Actions” and then select “download anyway”. The file is safe to download and is certainly not malicious.

Then, you will need to right click the .dll files and save them in the same “programs” folder. The files that you need to download and save are:

[wndchrm.exe](#)

[fftw3.dll](#)

[libfftw3-3.dll](#)

[libtiff3.dll](#)

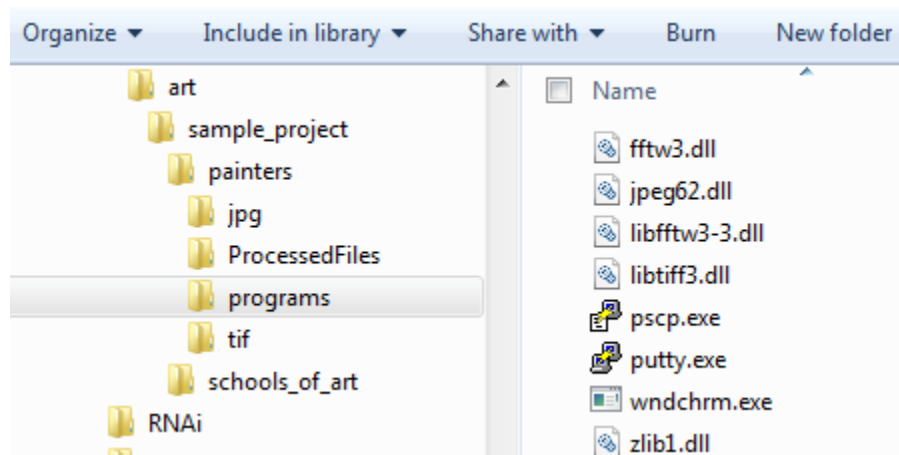
[jpeg62.dll](#)

[zlib1.dll](#)

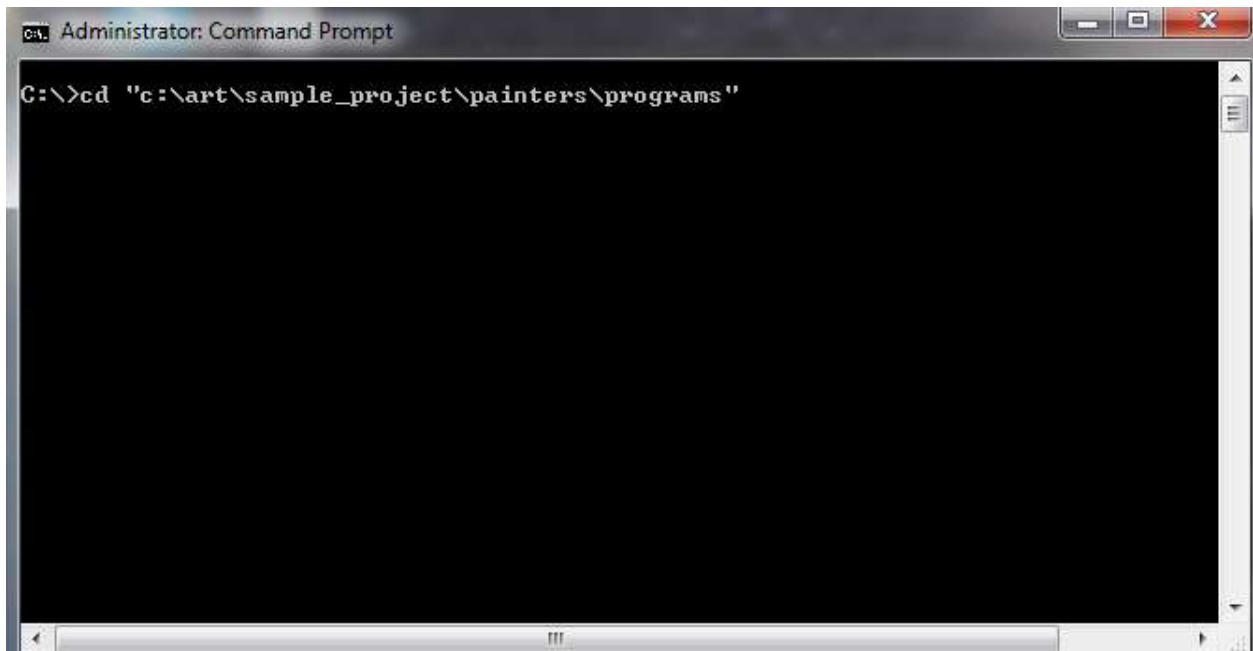
[libsndfile-1.dll](#)

What are DLLs? A Dynamic Link Library (DLL) is a file that contains machine code (programs). Applications that run on the computers can use that code (link to it) to perform common tasks that the DLL program does. The advantage of the DLL is that several different applications that need to perform the same task as part of their algorithm can use the same DLL, and that saves computer resources. Accessing the programs in the DLL is done through functions that the DLL *exports* to other applications.

When done, your “programs” folder should contain the Wndchrm files, in addition to the pscp.exe and putty.exe that we downloaded before.

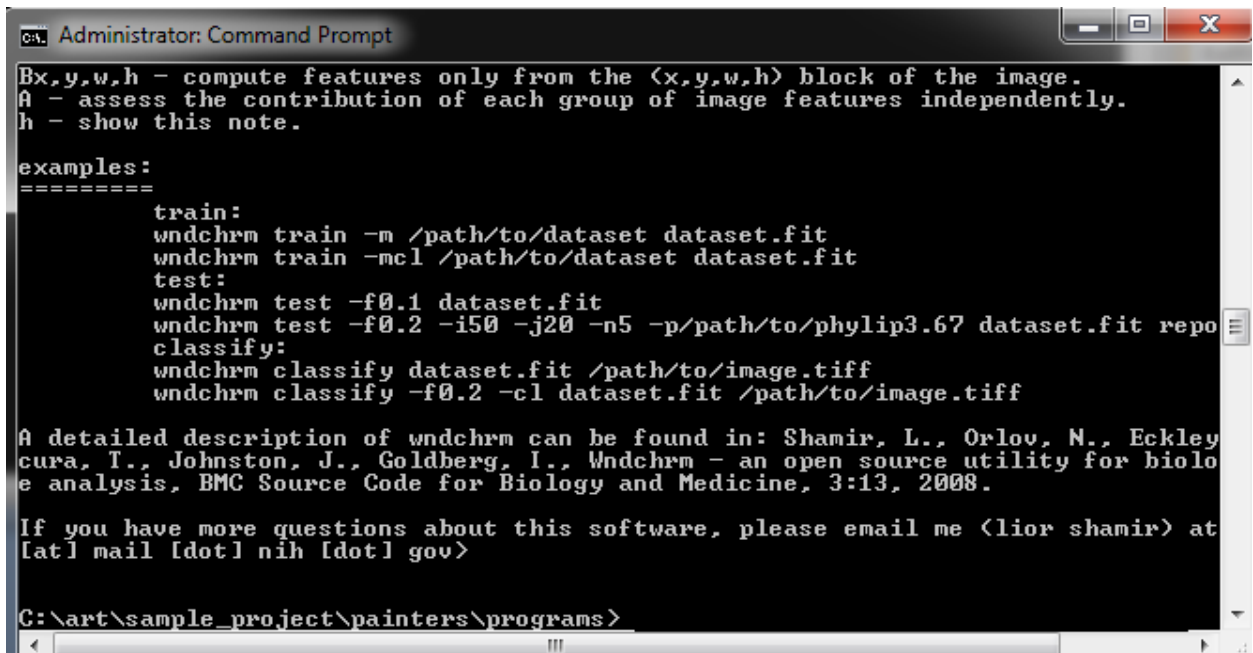


You can verify that Wndchrm works by launching the command prompt window, and changing the directory to the “programs” folder.



```
Administrator: Command Prompt
C:\>cd "c:\art\sample_project\painters\programs"
```

Then, type “wndchrn”. Wndchrn should display short instructions.



```
Administrator: Command Prompt
Bx,y,w,h - compute features only from the <x,y,w,h> block of the image.
A - assess the contribution of each group of image features independently.
h - show this note.

examples:
=====
train:
wndchrn train -m /path/to/dataset dataset.fit
wndchrn train -mcl /path/to/dataset dataset.fit
test:
wndchrn test -f0.1 dataset.fit
wndchrn test -f0.2 -i50 -j20 -n5 -p/path/to/phylip3.67 dataset.fit repo
classify:
wndchrn classify dataset.fit /path/to/image.tiff
wndchrn classify -f0.2 -cl dataset.fit /path/to/image.tiff

A detailed description of wndchrn can be found in: Shamir, L., Orlov, N., Eckley
cura, I., Johnston, J., Goldberg, I., Wndchrn - an open source utility for biolo
e analysis, BMC Source Code for Biology and Medicine, 3:13, 2008.

If you have more questions about this software, please email me <lior shamir> at
[at] mail [dot] nih [dot] gov>

C:\art\sample_project\painters\programs>
```

Now, we can run the experiment and create the phylogenies that reflect the similarities between the painters.

First, we need to train the machine learning algorithm. Given that the files that we downloaded from the server are located in the “ProcessedFiles” folder, the command line that will train the machine learning system will be as following:

```
wndchrm train -mlc -t4 c:\art\sample_project\painters\ProcessedFiles c:\art\sample_project\painters\ProcessedFiles\output.fit
```

After the image file names will be displayed on the screen, the file “output.fit” will be created in the “ProcessedFiles” folder.

Now, we can test how well the computer can differentiate between the different groups of paintings using the pattern recognition capabilities of Wndchrm.

What is pattern recognition? In the context of computer science, pattern recognition is the ability of the computer to find patterns in the data. In our experiment we look for patterns in the numerical values computed from each painting. Each class has 30 paintings in it, and the purpose of the pattern recognition algorithm is to find automatically the numerical values that are repetitive in paintings of that class, but not in the other classes. These patterns can make a distinction between one class and the other classes, and can be used to numerically characterize a certain artistic style from the other styles.

Using the pattern recognition capabilities of Wndchrm can be done by using the “test” command of *Wndchrm*:

```
wndchrm test -f0.15 -t4 -i25 -j5 -n20 -w c:\art\sample_project\painters\ProcessedFiles\output.fit
```

“-f0.15” tells *Wndchrm* to use 15% of the numerical content descriptors.

“-t4” tells *Wndchrm* that the image files are analyzed such that each file is divided to 4x4 tiles.

“-i25” and “-j5” means that 25 images from each class will be used for training and 5 will be used for testing.

“-n20” tells *Wndchrm* to repeat the experiment 20 times, such that in each run different images will be randomly allocated to training and test sets.

“-w” means that *Wndchrm* will use the Weighted Nearest Distance (WND) pattern recognition method.

When running, Wndchrm will show the files that are being classified, their class ID and their predicted class ID. If the class ID and the predicted class ID are the same, the classification of that painting is correct.

[illegible]

Wndchrm will test five images per class in each of the 20 runs, and when done will show the average classification accuracy of paintings. When done, it will show the classification accuracy.

.000000	monet	0.97222	1.03308	0.55373
	pollock	0.44606	0.41017	
.65049		1.00000	0.58842	0.34533
	renoir	0.68492	0.64420	
.66219		0.63019	1.00000	0.45094
	rothko	0.92317	0.97379	
.54954		0.49049	0.74865	1.00000
	vangogh	0.76013	0.71576	
.89404		0.89744	0.95162	0.54825

Accuracy: 0.542484

Average accuracy (20 splits): 0.524510

What is classification accuracy? The most basic way of measuring the performance of pattern recognition systems is the classification accuracy. From each class of images we allocate some images to training the classifier, so that the pattern recognition algorithm can identify repetitive patterns in the values of the image features computed from these images. The rest of the images are used for testing. Each test images is classified, and the prediction of the classifier is compared to the actual class that the painting belongs in. If the predicted class is the same as the actual class the classification is considered a hit, and if the two are different the classification is a miss. After classifying all test images, the classification accuracy is determined by the number of hits divided by the total number of classifications (the number of test images).

In the case above, Wndchrm was able to associate a painting to the class with average accuracy of 52.45%. The classification is done without knowing anything about the tested paintings other than the visual content, so the association of a painting to a class is done by analyzing the style of the painting alone. Any classification accuracy that is higher than random guessing is an indication that the machine learning system is informative in identifying paintings by their style. For instance, if we have 10 classes, classification accuracy higher than 10% indicates that the system is informative in the association of the paintings.

Now, we can change the number of features that we use by changing the value of the “-f” switch. For instance, to use 30% of the numerical image content descriptors we can use the following command line:

```
wndchrm test -f0.3 -t4 -i25 -j5 -n20 -w c:\art\sample_project\painters\ProcessedFiles\output.fit
```

We will need to run the same command with “-f0.05”, “-f0.1”, “-f0.15”, “-f0.3”, “-f0.4”, “-f0.5”. The amount of content descriptors that provides the highest classification accuracy will be the one used in our following experiments.

We can also ask Wndchrm to create a detailed report file showing the results of the experiment. That will be done by using the “-p” switch, and specifying another file in the end of the command line:

```
wndchrm test -f0.3 -t4 -i25 -j5 -n20 -w -p c:\art\sample_project\painters\ProcessedFiles\output.fit  
c:\art\sample_project\painters\ProcessedFiles\results.html
```

The file “results.html” will provide a detailed report on the experiment. To open the report file, just open the folder “ProcessedFiles” and double click the file “results.html”. The file will open in your default browser.

Here is an example of a report file of an experiment with nine painters:

622 Images in the dataset (tiles per image=16)

	dali	deChirico	ernst	kandinsky	monet	pollock	renoir	rothko	vangogh	total
Testing	17	17	17	17	17	17	17	17	17	153
Training	40	40	40	40	40	40	40	40	40	360

Results

Split 1	Accuracy: 0.71 of total (P=3.001755e-68) 0.71 ± 0.3 Avg per Class Correct of total Full details
Total	0.71 Avg per Class Correct of total Accuracy: 0.71 of total (P=0.000000e+00)

Confusion Matrix (sum of all splits)

	dali	deChirico	ernst	kandinsky	monet	pollock	renoir	rothko	vangogh
dali	13	1	2	1	0	0	0	0	0
deChirico	3	9	3	0	0	0	1	0	1
ernst	2	0	11	0	0	0	1	0	0

The upper part of the form shows the classes that were used in the experiment. In the specific report there are nine classes such that each class is a painter. The report specifies the names of the classes (in this cases the names of the painters), and the number of training and test images for each class. Note that the number of training and test images are the same for all classes. According to the report, the classification accuracy of a painting to a painter is 0.71, or 71%.

What is a confusion matrix? A confusion matrix is a visualization of the results of a supervised machine learning experiment. The rows of the matrix represent the actual classes that the sample belong in, and the columns represent the predicted classes. The values in the cells are the numbers of the images from each class (the rows) classified to each of the other classes (the columns). If the classifier is always accurate, all values will be on the diagonal.

Below the summary of the classification accuracy you will find the confusion matrix. The confusion matrix reflects how well the computer analysis can associate paintings to their classes. For instance, the following confusion matrix shows the classifications of paintings of the nine painters:

Confusion Matrix (sum of all splits)

	dali	deChirico	ernst	kandinsky	monet	pollock	renoir	rothko	vangogh
dali	13	1	2	1	0	0	0	0	0
deChirico	3	9	3	0	0	0	1	0	1
ernst	3	2	11	0	0	0	1	0	0
kandinsky	0	0	1	12	0	0	2	0	2
monet	0	0	0	1	11	0	1	0	4
pollock	0	0	0	1	0	15	0	0	1
renoir	0	0	0	1	0	0	14	0	2
rothko	0	0	0	0	0	0	0	17	0
vangogh	0	0	1	1	4	4	0	0	7

For instance, the confusion matrix shows that out of 17 Salvador Dali paintings, 13 were classified correctly as paintings of Dali, one was classified as a painting of de Chirico, two as paintings of Max Ernst, and one as a painting of Kandinsky. The confusion matrix therefore reflects the confusion of the algorithm when classifying the paintings.

The similarity matrix shows the similarities between the different painters. These similarity numbers are computed by the algorithm, and reflect the similarities between the different classes. The similarity values are between 0 and 1, such that 1 means full similarity. In this example, each class is a painter and therefore the similarity matrix reflects the similarity between painters.

What is a similarity matrix? The similarity matrix visualizes the differences between the classes as deduced by the pattern recognition algorithm. The cell i,j shows the similarity between class i and class j . The values are normally in the range of $[0,1]$. The diagonal shows the similarity of the class to itself, which is normalized to 1.

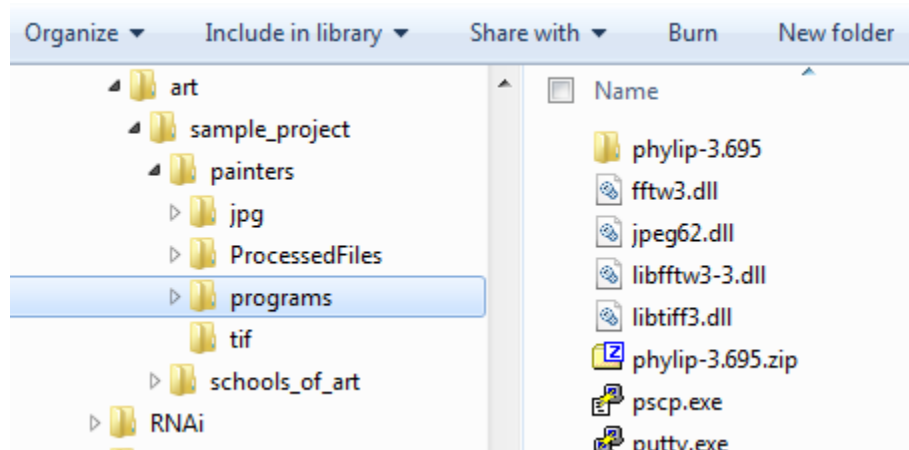
Average Similarity Matrix

	dali	deChirico	ernst	kandinsky	monet	pollock	renoir	rothko	vangogh
dali	1.00	0.97	0.97	0.94	0.85	0.80	0.94	0.90	0.92
deChirico	0.97	1.00	0.98	0.94	0.84	0.79	0.95	0.92	0.93
ernst	0.94	0.94	1.00	0.92	0.85	0.78	0.93	0.87	0.92
kandinsky	0.94	0.92	0.94	1.00	0.89	0.84	0.96	0.87	0.94
monet	0.87	0.85	0.89	0.91	1.00	0.94	0.95	0.76	0.98
pollock	0.54	0.55	0.60	0.58	0.69	1.00	0.67	0.45	0.79
renoir	0.93	0.92	0.93	0.95	0.94	0.87	1.00	0.84	0.96
rothko	0.88	0.86	0.87	0.82	0.68	0.62	0.76	1.00	0.75
vangogh	0.82	0.84	0.87	0.84	0.98	0.99	0.93	0.73	1.00

Our last step will be the visualization of the similarity matrix as a phylogeny (evolutionary tree). A phylogeny is a method for visualization of similarities between data points. The phylogeny will be created by using another software tool called *Phylip*. Traditionally, phylogenies have been used for visualizing the similarities between organisms based on their genomes, but in our experiment we will use *Phylip* to visualize similarities between artistic styles. *Phylip* can be obtained from its download web page at <http://evolution.genetics.washington.edu/phylip/getme.html>

What is a phylogeny? A phylogeny (evolutionary tree) is a visualization of similarities between organisms, and shows the common descents of different organisms. It shows the ancestor/descendent relationships of different organisms in the form of a tree.

In the download page, find the Windows download zip file “Zip archive of PHYLIP 3.695 for Windows”, and download it to your “programs” folder. Then, right click the file to extract the files inside the zip file. A new folder called “phylip-3.695” will appear under your “programs” folder.



Now we can create the phylogeny of the classes in our experiment by using the command line of the form:

```
wndchrm test -f0.3 -t4 -i25 -j5 -n20 -w -p[path to phylip] [path to fit file] [path to report file]
```

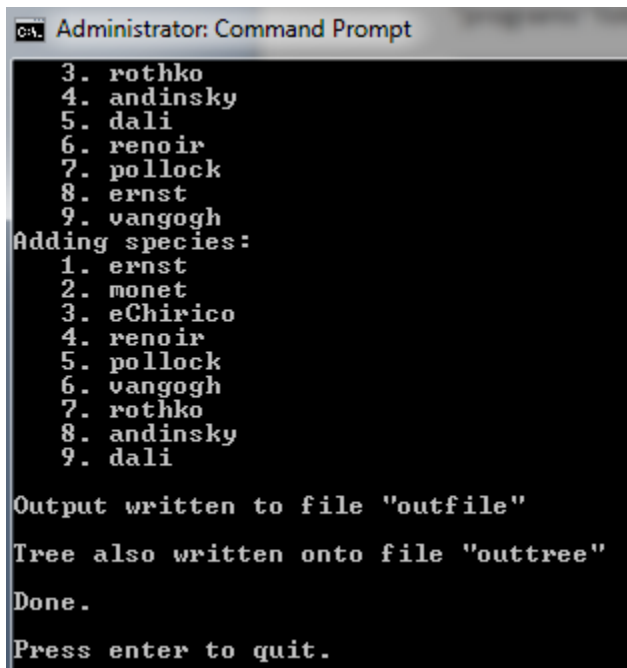
Where instead if “[path to phylip]” we need to specify the path to the folder where *Phylip* is (in our case it is the “programs” folder), [path to fit file] is the path to the output file that we created, and [path to report file] is the path to the HTML report file. For instance, the command can be the following:

```
wndchrm test -f0.3 -t4 -i25 -j5 -n20 -w -pc:\art\sample_project\painters\ programs\phylip-3.695
c:\art\sample_project\painters\ProcessedFiles\output.fit c:\art\sample_project\painters\ProcessedFiles\results.html
```

As in the previous experiment, the paths should be changed based on the paths in your computer. Note that the drive letter in the path to phylip (“pc:\art\sample_project\painters\ programs\phylip-3.695” in the above example) should be non-capital.

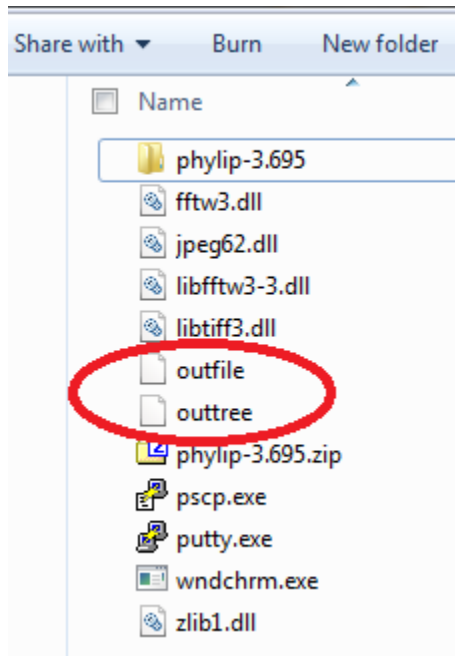
As you can see, after the “-p” switch we specified the path to *Phylip*. All *Phylip* files are located in the “exe” subfolder of the “phylip-3.695” folder. Now, you can double-click the file “results.html”. The report will also show you the phylogeny of the classes.

The command-line window will show the files that are being classified (as in the previous experiment), but in the end of the run will also show some more information such as the following:

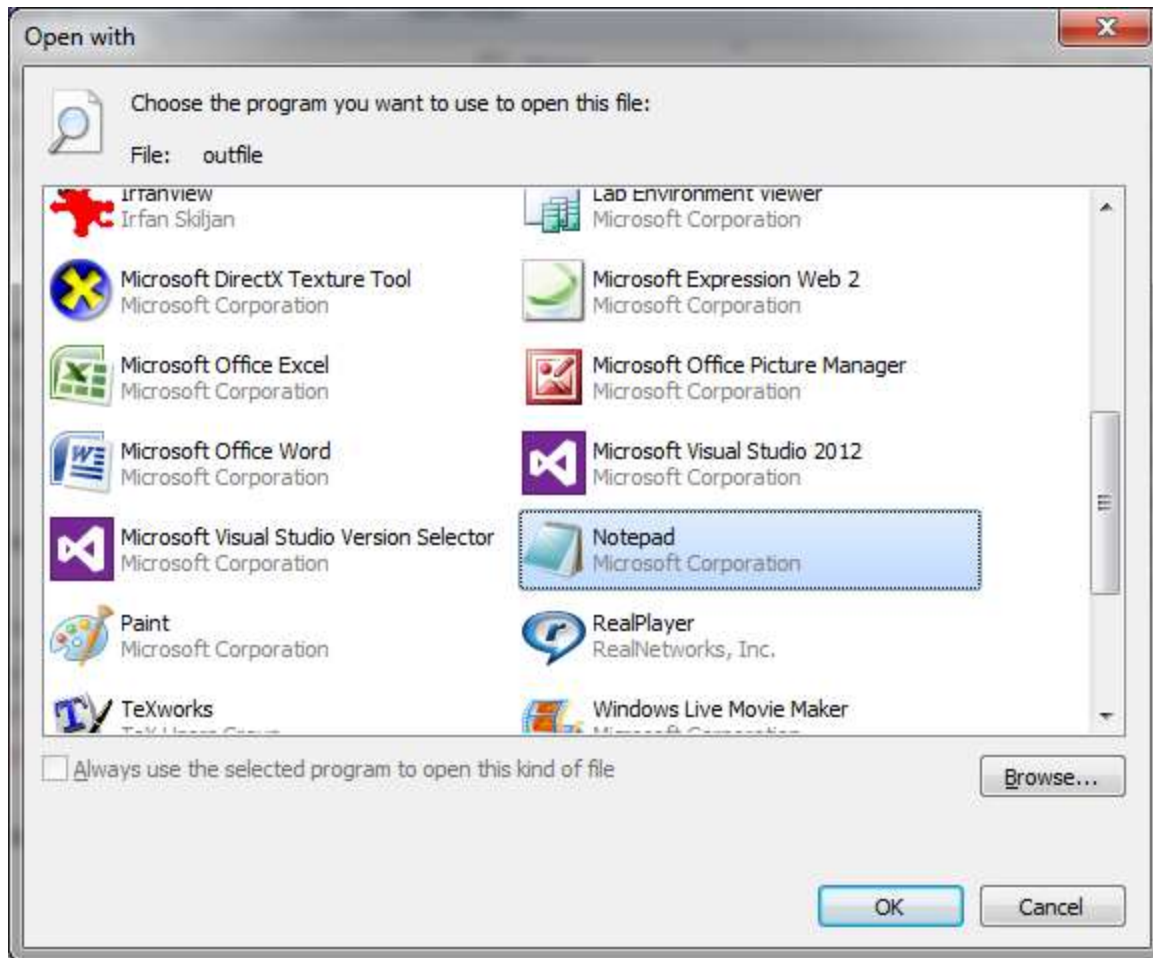


```
Administrator: Command Prompt
3. rothko
4. andinsky
5. dali
6. renoir
7. pollock
8. ernst
9. vangogh
Adding species:
1. ernst
2. monet
3. eChirico
4. renoir
5. pollock
6. vangogh
7. rothko
8. andinsky
9. dali
Output written to file "outfile"
Tree also written onto file "outtree"
Done.
Press enter to quit.
```

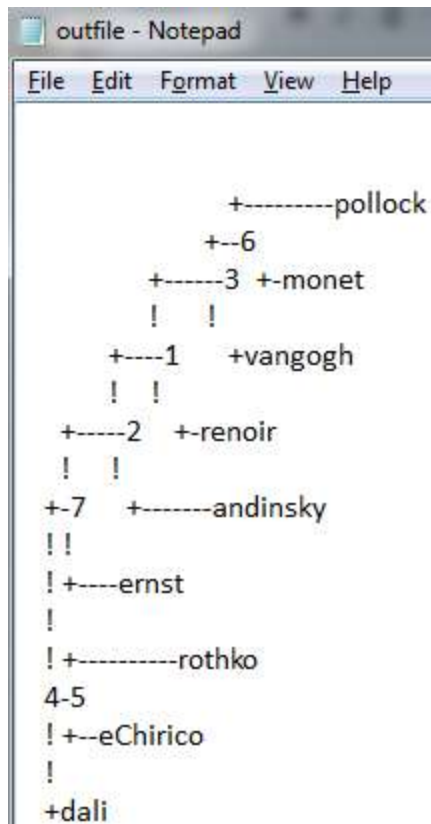
That information is the construction of the similarity tree. When done, you will find two files in the “programs” folder. One is called “outtree” and the other is called “outfile”.



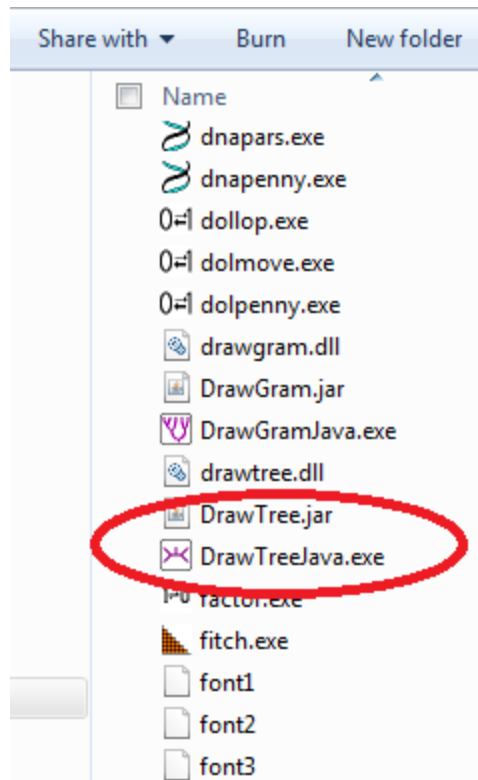
“outfile” contains information about the tree of similarity. You can open that file by double-clicking it and then choosing “Notepad” as the program to open the file with.



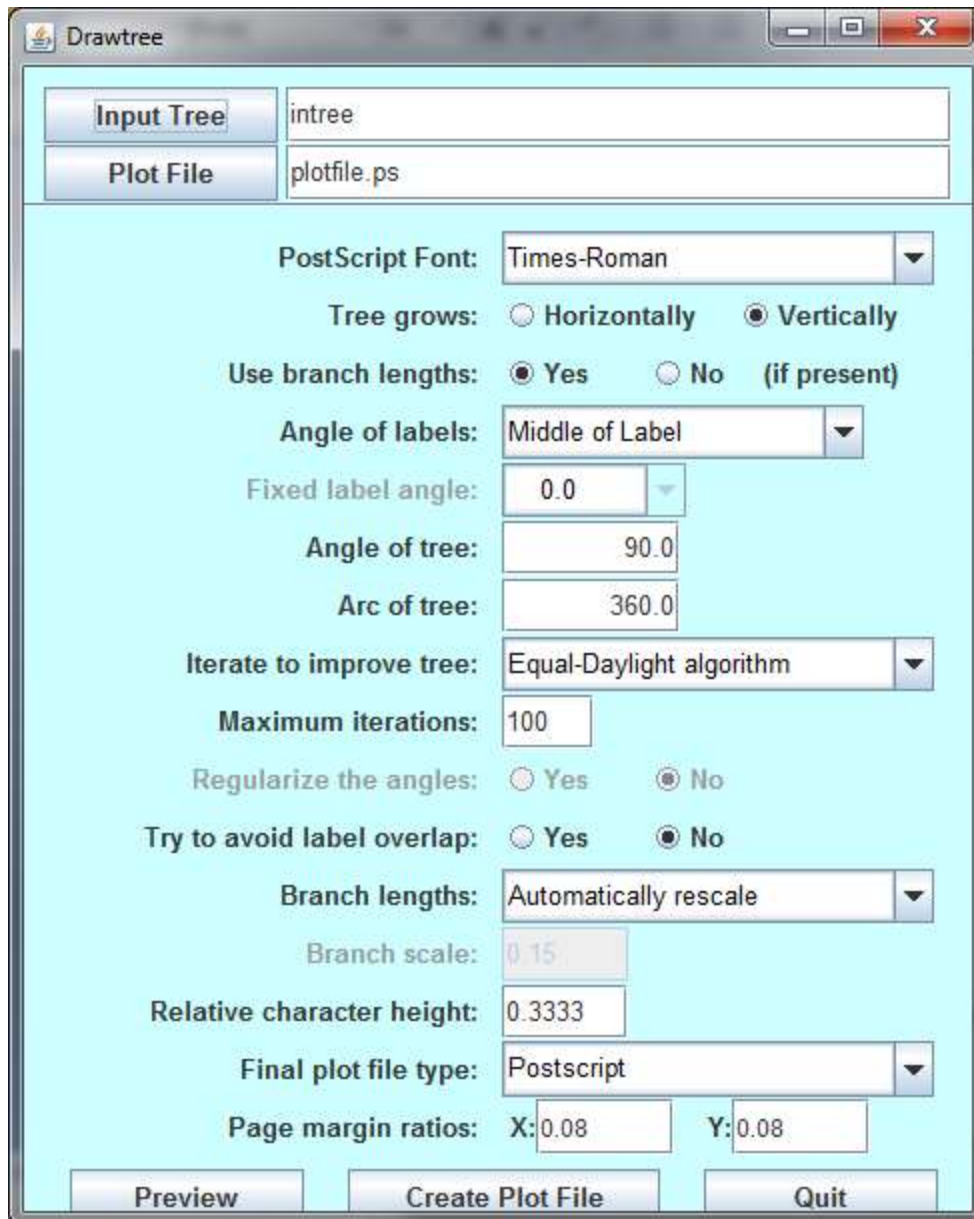
The file will open in Notepad, and will show the basic form of the tree.



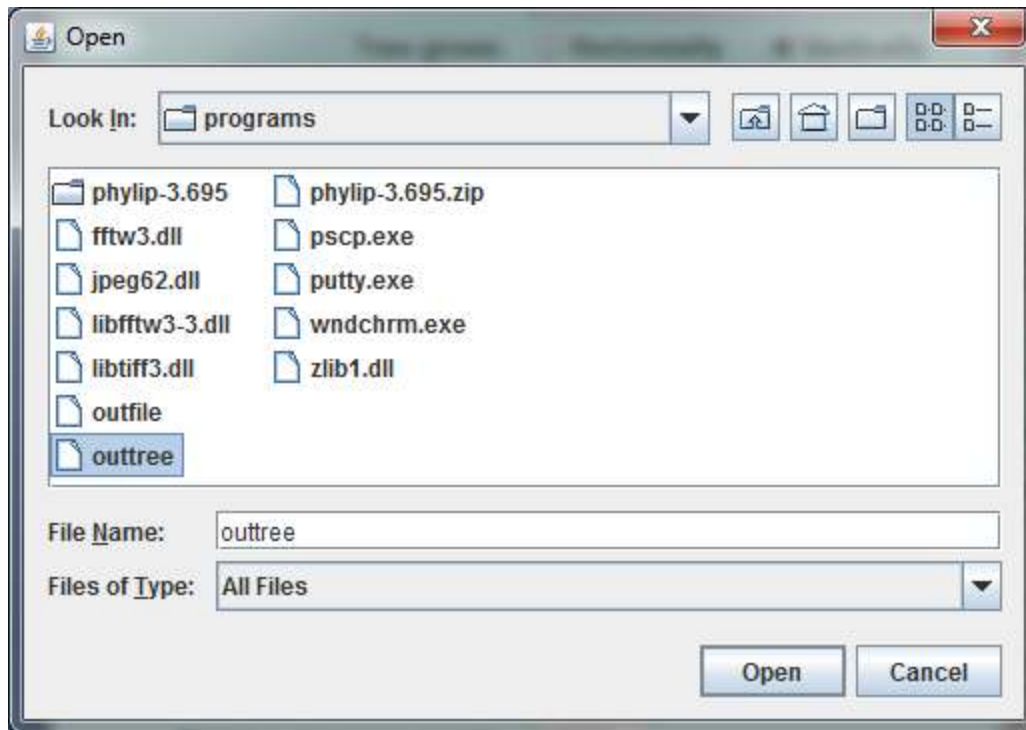
To create a graphic form of the phylogeny, we need to use another program called "DrawTree". The program is in the "Phylip/exe" folder.



Running the program will open the following window:

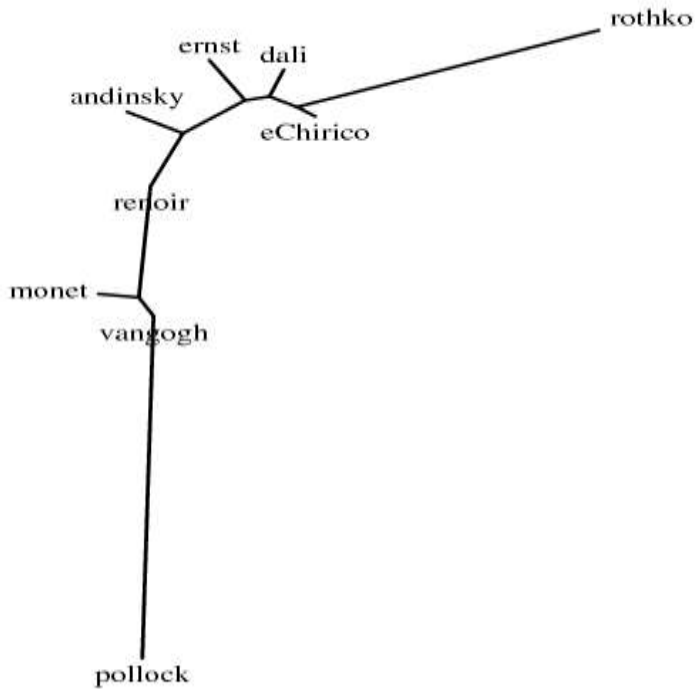


To select the input file, we need to click on the “Input Tree” button, and then select the file “outtree” that was created in the end of the previous step. The file will be located in the “programs” folder. Then click “Open”.



Then you can click the “Preview” button to see the tree, and then on “Create Plot File” to save the tree to a file.

Here is a sample phylogeny created with paintings from nine artists of three different schools of art:



Phylip only shows labels that are eight letters or less so some of the artist names are not the full names.

8. Enabling algorithms

This section briefly describes the algorithms that enable the image analysis. The description is taken from (Shamir & Tarakhovsky, 2012). The paper is available at: <http://dl.acm.org/citation.cfm?id=2307726>. A more detailed technical description is available at (Shamir et al., 2010, <http://dl.acm.org/citation.cfm?id=1670672>).

The image analysis method is based on the WND-CHARM scheme (Shamir, 2008; Shamir et al., 2008), which was originally developed for biomedical image analysis (Shamir et al., 2008; Shamir et al., 2009). The CHARM (Shamir, 2008; Shamir et al., 2010) set of numerical image content descriptors is a comprehensive set of 4027 features that reflect very many aspects of the visual content such as shapes (Euler number, Otsu binary object statistics), textures (Haralick, Tamura), edges (Prewitt gradient statistics), colors (Shamir, 2006), statistical distribution of the pixel intensities (Multiscale histograms, first four moments), fractal features (Wu, Chen, Hsieh, 1992), and polynomial decomposition of the

image (Chebyshev statistics). These content descriptors are described more thoroughly in (Shamir, 2008; Shamir et al., 2008; Shamir et al., 2009; Shamir et al., 2010). This scheme of numerical image content descriptors was originally developed for complex morphological analysis of biomedical imaging, but was also found useful for the analysis of visual art (Shamir et al., 2010; Shamir, 2012).

An important feature of the set of numerical image content descriptors is that the color descriptors are based on a first step of classifying each pixel into one of 10 color classes based on a fuzzy logic model that mimics the human intuition of colors (Shamir, 2006). This transformation to basic color classes ensures that further analysis of the color information is not sensitive to specific pigments that were not available to some of the classical painters in the dataset, or to the condition and restoration of some of the older paintings used in this study.

Once numerical image content descriptors are computed, each feature is assigned with a Fisher discriminant score, as defined by Equation 1

$$\text{Equation (1)} \quad W_f = \frac{\sum_{c=1}^N (\overline{T_f} - \overline{T_{f,c}})^2}{\sum_{c=1}^N \sigma_{f,c}^2} \cdot \frac{N}{N-1},$$

where W_f is the weight assigned to the feature f , $\overline{T_f}$ is the mean of the values of feature f in the entire dataset, $\overline{T_{f,c}}$ is the mean of the values of feature f in the class c , and $\sigma_{f,c}^2$ is the variance of feature f in the images of class c . Fisher discriminant scores can be conceptualized by the variance of the values of a certain feature across the image dataset, divided by the mean of the variances of that feature within the classes.

Then, the image features are ranked by their Fisher discriminant scores, and the 50% of the image features with the lower Fisher scores are rejected. The distance $d_{I,P}$ between each image I to each painter P is then measured by using Weighted Nearest Distance method, with the Fisher scores used as weights [Shamir et al., 2008] as defined by Equation 2

$$\text{Equation (2)} \quad d_{I,P} = \frac{\sum_{m \in P} \left| \sum_{f=1}^{|I|} W_f (i_f - m_f)^2 \right|^k}{|P|},$$

where P is the training images of painter p , i is the vector of numerical image content descriptors extracted from the image I , W_f is the Fisher discriminant score of feature f as defined by Equation 1, and

the exponent k is a constant set to -5. The -5 value was determined empirically, and is thoroughly discussed in (Orlov et al., 2008).

The distance $d_{I,P}$ can be used to compute the similarity $S_{I,P}$ between the image I and painter P by using Equation 3

$$\text{Equation (3)} \quad S_{I,P} = \frac{1}{d_{I,P} \cdot \sum_{j=1}^N \frac{1}{d_{I,j}}},$$

where N is the total number of painters in the dataset.

The arithmetic mean of the similarity values computed by Equation 3 across all images of a certain pair of painters provides the quantified visual similarity between these two painters (Shamir et al., 2008; Shamir et al., 2010), and computing the visual similarities between all pairs of painters in the dataset produces a similarity matrix in which the cell i,j is the measured similarity between painter i and painter j . The similarity matrix is then visualized as a phylogeny using the open source Phylip software package, which was originally developed to visualize the lines of descent among different organisms, but can be used to visualize any matrix of similarity as a tree. The source code used for the analysis is open and publicly available at <http://people.cs.ksu.edu/~lshamir/downloads/ImageClassifier/>, and the source code and further description of the phylip package is available at <http://evolution.genetics.washington.edu/phylip.html>.

Bibliography

Gariff, D. Denker, E., Weller, D.P. 2009. The world's most influential painters and the artists they inspired. Barron's Educational Series, Hauppauge, NY.

O'Mahony, M. 2006. World Art. The essential illustrated history. p. 87.

Orlov, N., Shamir, L., Macura, T., Johnston, J., Eckley, D. M., and Goldberg, I. 2008. WND-CHARM: Multi-purpose image classification using compound image transforms. *Pattern Recognition Letters*, 29, 1684-1693.

Shamir, L. 2006. Human perception-based color segmentation using Fuzzy Logic. *International Conference on Image Processing Computer Vision and Pattern Recognition*, 2, 496-505.

Shamir, L. 2008. Evaluation of face datasets as tools for assessing the performance of face recognition methods. *Intl. J. Comp. Vision*, 79, 225-229.

Shamir, L. et al. 2008. Wndchrm - an open source utility for biological image analysis. *BMC - Source Code for Biology and Medicine* 3, 13.

Shamir, L. et al. 2009. Knee X-ray image analysis method for automated detection of Osteoarthritis. *IEEE Trans. on Biomed. Eng.*, 56, 407-415.

Shamir, L. et al., 2010. Impressionism, expressionism, surrealism: Automated recognition of painters and schools of art. *ACM Transactions on Applied Perception*, 7, 8.

Shamir, L. 2011. Assessing the efficacy of low-level image content descriptors for computer-based fluorescence microscopy image analysis, *Journal of Microscopy*, 243, 284-292.

Shamir, L. 2012. Computer analysis reveals similarities between the artistic styles of Van Gogh and Pollock. *Leonardo*, 45, 2, 149-154.

Shamir, L., Tarakhovsky, J., 2012. Computer analysis of art, *ACM Journal on Computing and Cultural Heritage*, 5(2), 7.

Walther, I., Metzger, R. 2006. Vincent van Gogh: The complete paintings. Taschen Verlag, Cologne, Germany.

Wu, C. M., Chen, Y. C., Hsieh, K.S. 1992. Texture features for classification of ultrasonic liver images, *IEEE Trans. Medical on Imaging* 11, 141-152.