

## Colorization GUI

The colorization process uses an open source implementation of the algorithm that was proposed by Anat Levin, Dani Lischinski and Yair Weiss at the School of Computer Science and Engineering, the Hebrew University of Jerusalem. (see <http://www.cs.huji.ac.il/~yweiss/Colorization/>)



By scribbling (marking) areas with different colors, the algorithm calculates the color for the complete image. Marked areas are turned into the given color.

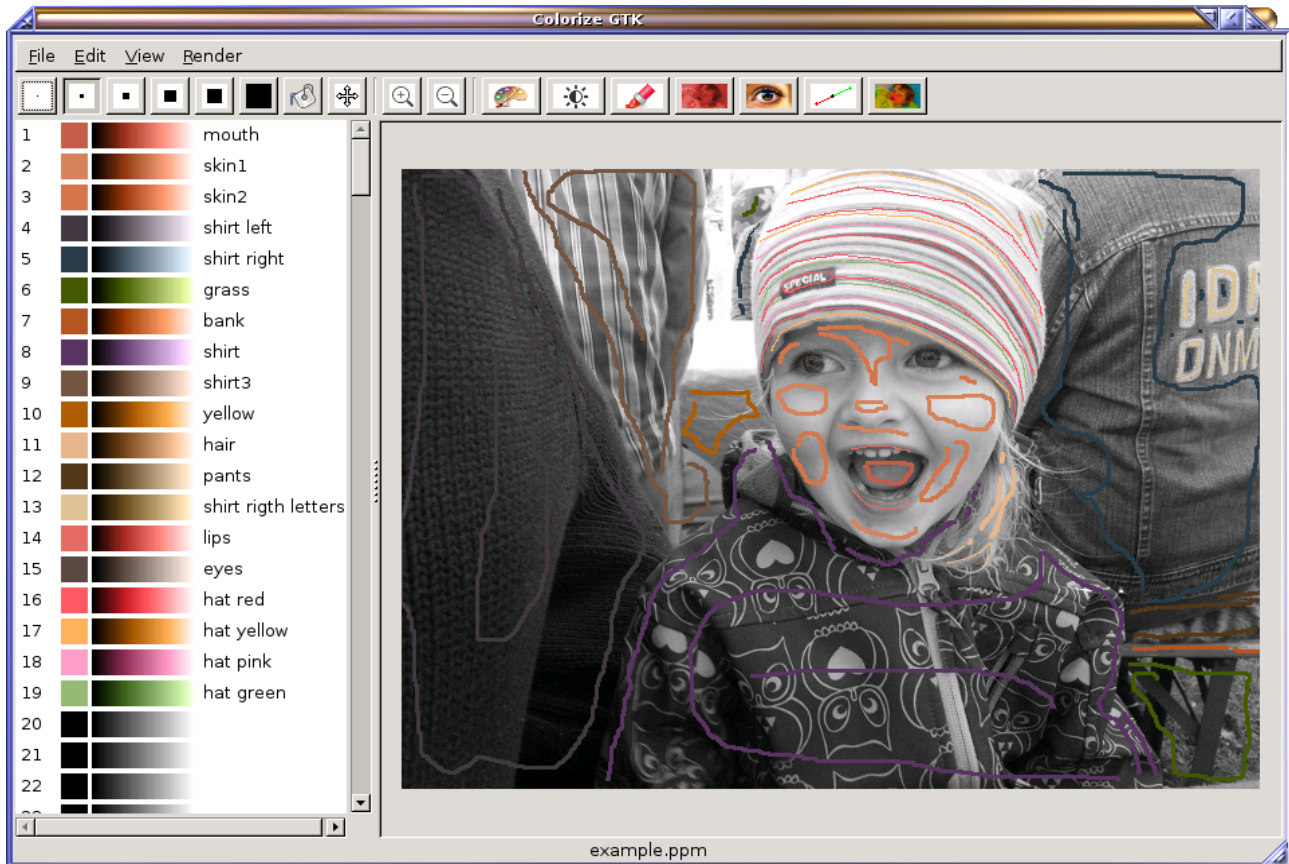
The colorization process will apply colors to neighbor pixels of the marked areas. The color at the neighbor pixels is similar, if the brightness of that pixels are similar. Between marked areas, the color is fading automatically using a weighting function. If the edge between areas is soft, the color change is also soft.

The colorize GUI simplifies the process of adding color to gray images. Where other (much more advanced graphics editor) can do the same, this GUI provides some important features that make things allot easier:


- Marked pixels are not anti-aliased, which is required for the algorithm to work correctly.
- The marked pixels are stored in a separate file, keeping the original image untouched.
- The marked pixels are stored as indexed byte map with palette, so colors can be changed without redrawing marked areas.
- Current selected color can be highlighted (red) for better view on an image with similar brightness.
- A colorize button is used to render the result without using the command line tool.
- Color images can be recolored. Areas can be marked without changing the color, changing the color or removing the color.
- The GUI can handle image sequences and provides a time line with key-frames and marked images.


## Still image


To colorize a gray image, areas need to be marked with target color. The colorization algorithm then fills the marked areas with this color and blends them on the edges between areas. As you can see at the edge of the hair and the skin, the color is blended with the change in brightness of the gray image.




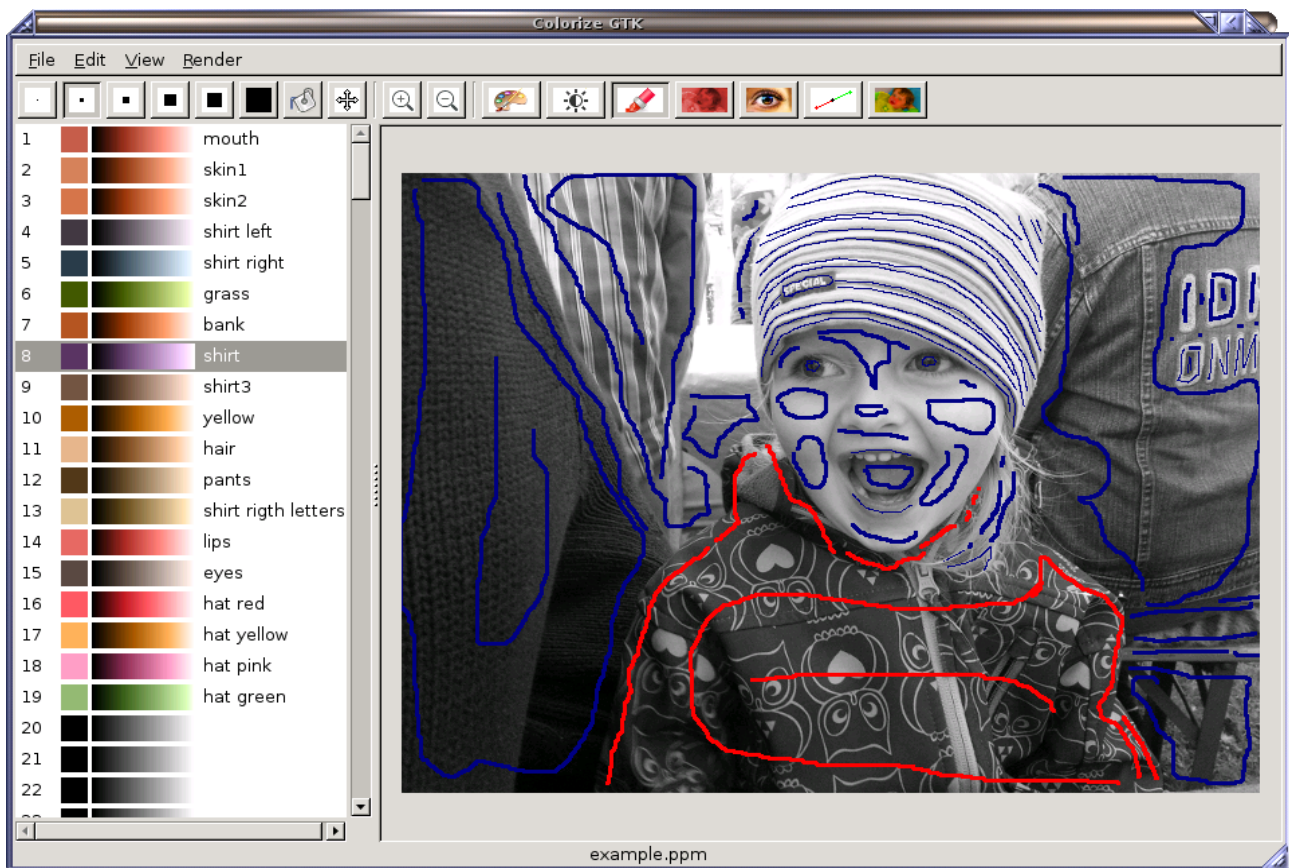
To get started, run `colorize_gtk` and open a file select “File → Open” from the menu and then select “example.ppm” from “examples” directory of source archive. The GUI will also load “example.ppm\_marked” file and the “example.ppm\_palette” file, if exist.

To colorize the displayed image, click on  at the toolbar. After some seconds, a window with the colorized result will open. From that window the result can be saved.

To mark the image with a color, select the color from the palette (left of the window). If you like to change the color or define a new one, double-click on the name and enter it's name and click on  at the toolbar to change it's color. Note that the value of the color is not important, because it is taken from the gray image during colorization process. The fading bars at the palette will show the colorization result of the color at different gray level.

Now select your desired mark pen size by selecting  tool at the toolbar. If you use left mouse button, it will draw, if you use right mouse button it will erase areas. Click on the image to mark, select “Edit → Undo” from the menu, to revert the line you just drew. Note that the gray image is not touched, so scribbling will not alter it. An extra mark file “example.ppm\_marked” stored all marked lines. You may erase all marked colors or only the selected color at the “Edit” menu. The gray image will show up without the marked lines that has been removed.

In order to view only the selected color, click on  at the toolbar. The selected color will be shown in red, all other color in blue. Click again and the color are shown again.

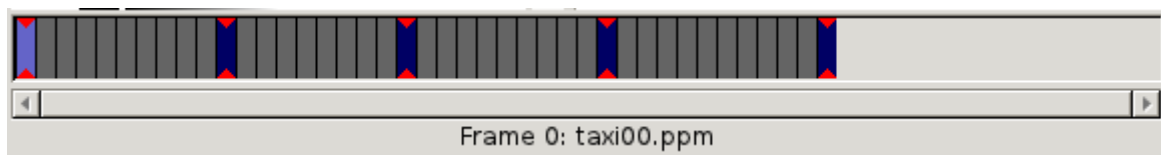


This is quite helpful if the marked color has only little hue, and so it might look similar to the gray image in the background.



## Film Sequence

Select “File → Open” from the menu and then select “taxi00.ppm” from “examples/taxi” directory of source archive. You will see a time-line below the image:



When open one image of a sequence, other images are automatically detected and displayed in a time-line. The bright bar in the time-line is the current selected image. Blue bars indicate that there are marked pixels. The red triangles indicate key frames. In this example all marked images are key-frames too.

Click on the time-line to change selected image or use cursor keys (left / right) and “Home” / “End” keys to change selected image. To jump directly to marked images, use SHIFT + cursor keys.



Whenever you change the current image, marked pixels are saved automatically to an extra file. (e.g. taxi00.ppm\_marked). If all marked pixels are erased, the extra file would be removed.

To set or unset key-frames, select a frame and press CTRL+K or select “Edit → Keyframe” from the menu. Information about all set key-frames is store in an extra file “sequence”.

All images of a sequence share the same palette. The palette information is stored in an extra file “palette”.

When marking several frames, it can be faster to copy and paste marked color or all colors from one frame to another. In this example only three cars move, so only minor change must be done. In order to copy one or all colors from one frame to another, select “Edit → Copy” or “Edit → Copy All” from menu, select target frame from the time-line and select “Edit → Paste”. If there are already marked pixels, select “Edit → Clear” first to remove them before paste.



Now you can manually correct the marked pixels on the cars using the erase tool  and the draw tool  from the toolbar.

## Key-frames and Rendering


Key-frame help to reduce the number of frame that are colorized at a time. Rendering complete film sequences would consume too much memory, especially with higher resolution than our “Hamburg Taxi Sequence” example. When using key-frames, only the sequences between key-frames (including the key-frames itself) are colorized at a time. It makes sense that the key-frames are also marked frames, so the colorization algorithm has at least two marked frames. On fast moving objects it makes sense to add additional marked frames that help the colorization algorithm to follow these objects. These additional marked frames must mark the complete scene, only the moving objects and the area around them is required. In our example, cars are moving slow, so there are no additional marked frames required.

In order to render a sequence, enter the directory where the images, the marked images, the sequence and the palette is stored. In our example, use a shell and enter the directory “examples/taxi/” of the source archive. Now enter “colorize sequence list”:

```
$ colorize sequence list
Got 11 frames from sequence (frames 0..10)
Got 10 frames from sequence (frames 10..19)
Got 11 frames from sequence (frames 19..29)
Got 12 frames from sequence (frames 29..40)
$
```

As you can see, all sequences between key-frames (including the key-frames itself) are only 10-12 frames, so the colorization algorithm only needs to render these number of frames at a time. To render the sequence enter “colorize sequence”:

```
$ colorize sequence
Got 11 frames from sequence (frames 0..10)
  11 frames, please wait...
Elapsed time: 1 minutes, 38 seconds
Got 10 frames from sequence (frames 10..19)
Colorizing 10 frames, please wait...
Elapsed time: 1 minutes, 29 seconds
Got 11 frames from sequence (frames 19..29)
Colorizing 11 frames, please wait...
Elapsed time: 1 minutes, 40 seconds
Got 12 frames from sequence (frames 29..40)
Colorizing 12 frames, please wait...
Elapsed time: 1 minutes, 47 seconds
$
```

Now each gray frame (“taxi00.ppm”, “taxi01.ppm”, ...) is rendered and stored in a color frame (“colorized\_taxi00.ppm”, “colorized\_taxi01.ppm”, ...). Click on  at the toolbar to see the colorized frames. You can now scroll through the time-line and verify the results.

## Changing or removing color



The same process for colorization can be used to change color. In order to change a color, areas are marked with a new color. To keep original color around these areas, they have to be marked with white color where all components need to have a value of 255.

In order to turn areas from color into gray, they can be marked with any gray color except white.

## YUV space and levels

Most colorization software is using YUV space. The component Y (lightness) is taken from gray image and UV color vector is rendered during the colorization process. After colorization, the YUV space is converted to RGB and then displayed or stored.

Select “Edit → Adjust levels” from menu to change the YUV ↔ RGB conversion process.

### Black Level:

Most images and film sequences have black levels above 0. If some part of the image shows something completely black or dark, it does not mean that the pixel level is also 0. If the black level would be lower than the actual black parts of the image, the colorization would also apply color to black, which causes a “chocolate effect”, that is known from old colorized post cards.



*Black Level at 0*



*Black Level at 50*

To find out the actual black level, use the palette's pick tool and hold down left mouse button while sliding over black parts of the image. Watch the RGB values at the palette window. Use the lowest level as black level.

### White Level:

Similar to the black level, the white parts of the image do not necessarily have a brightest pixel value of 255. When the picture gets overexposed, the maximum brightness is reached. Using a value that is higher will cause overexposed parts to be colorized, which causes unnatural effects like “blue clouds”.



*White Level at 255*



*White Level at 240*

Find the white level using the pick tool as described above. Use the slider to select the brightness of

overexposed image parts.

#### Scale Level:

Normally the original black and white levels are preserved after colorization. If this box is checked, the range between black and white levels is scaled to full range from 0 to 255. Use this button only for testing, use an image processing software to correct black and white levels after colorization.

#### Fade Level and Use modified YUV:

The original UV vector length is affected by saturation (more color) and brightness (more light). Darker regions of an image have lower UV vector length than brighter regions of the same color and saturation. Most colorization software do not alter UV vector length on different gray levels of the original image. The result is high saturation on dark regions (similar to “chocolate effect”) and low saturation on bright regions. A better approach is to use an UV vector that only represents color and saturation. The result is a constant saturation on all gray levels of the original object. If regions of the image are overexposed, the saturation is only reduced as required to prevent false colors.



*Regular YUV model (Black Level 33)*



*Modified YUV model (Black Level 33)*

Use the “Fade Level” slider to reduce the high saturation on dark pixels, or enable the “Use modified YUV” box to change to a modified YUV space. The “Fade Level” slider becomes obsolete in this case.