

Ripping subtitles from video files using SubRip

Written by *ai4spam*. Last modified on Friday, June 17, 2005 3:02 AM by *ai4spam*.

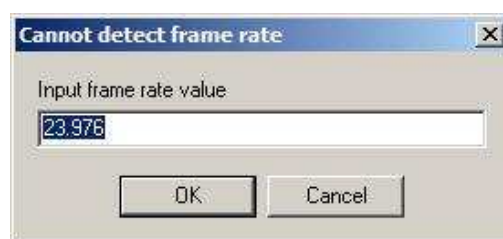
Please post your comments, suggestions and questions in the guestbook at the bottom of this page.

Some video files have subtitles "burned into them". SubRip can be used to extract the subtitles as text, as well as save them as bitmaps for later removal. This guide shows you how to extract the subtitles.

Open the video file by clicking on the button encircled in *red* below, or selecting *Open Hard Subbed Video files* from the *File* menu:



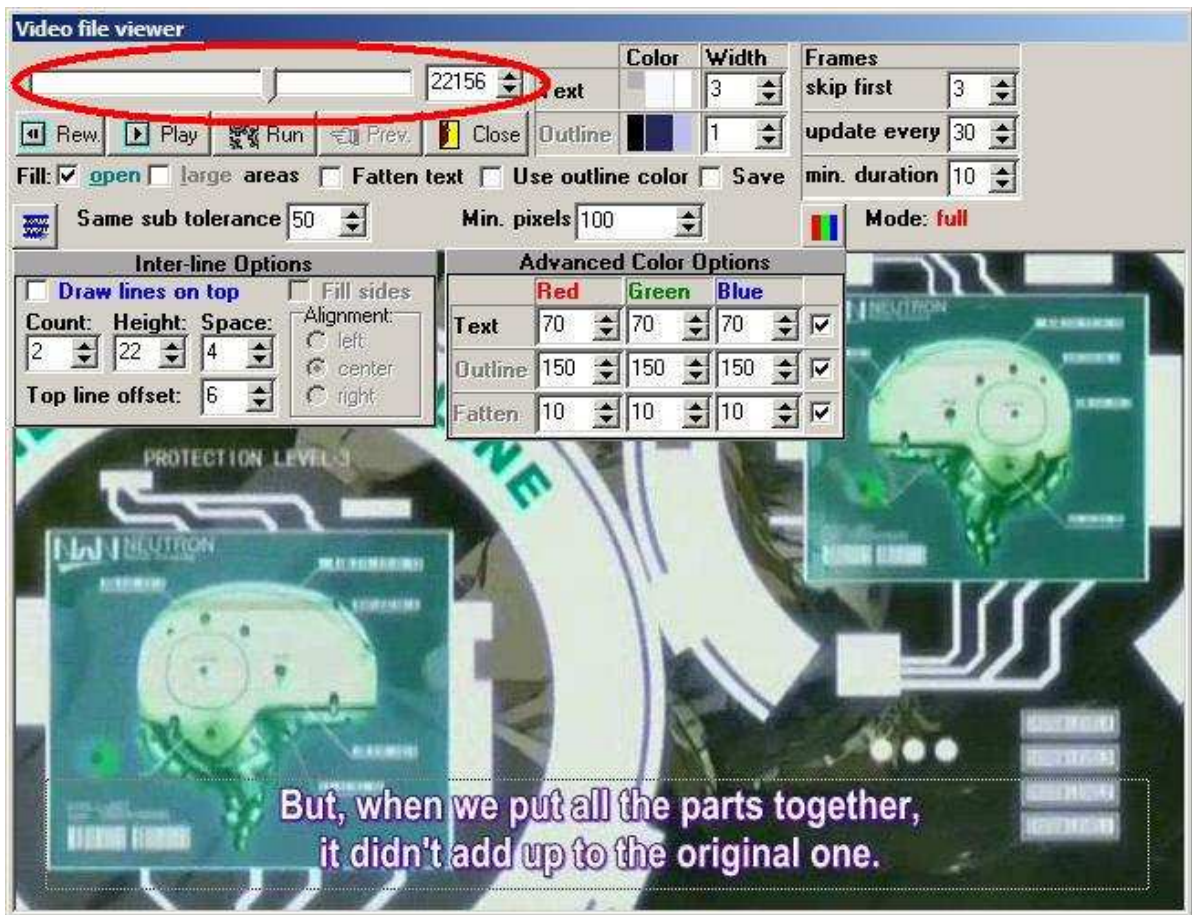
SubRip will try to open any file that [AviSynth](#) supports, but it can only detect the frame rate of .avi files. If you open another kind of file, SubRip will ask you to input the frame rate:



Please note that, in addition to [AviSynth](#), you also need to have the appropriate codecs and filters installed. For example, to open DVDs, you need [DGMPGDec](#). SubRip creates a file named "temp.avs" in its directory. The rule of thumb is: if you cannot play that file in your favorite media player, then neither will SubRip be able to open it. Also, some codecs and filters do not provide the ability to seek to an arbitrary frame. Normally, SubRip only moves forward, but it occasionally needs to seek to the first frame of a subtitle after changing the detection settings.

[AviSynth](#) seems to have its own buffering, but only between the previous and next keyframes. If you notice that seeking is inaccurate in some particular video file, the best approach is to convert it to an .avi file.

The *Video file viewer* window opens. Use the track bar or the edit box encircled in *red* below to move to a frame where you can see subtitles. Alternatively, press the *Play* button and let the video play, then Press the *Pause* button to stop the video when you see a subtitle, preferably with two or more lines.

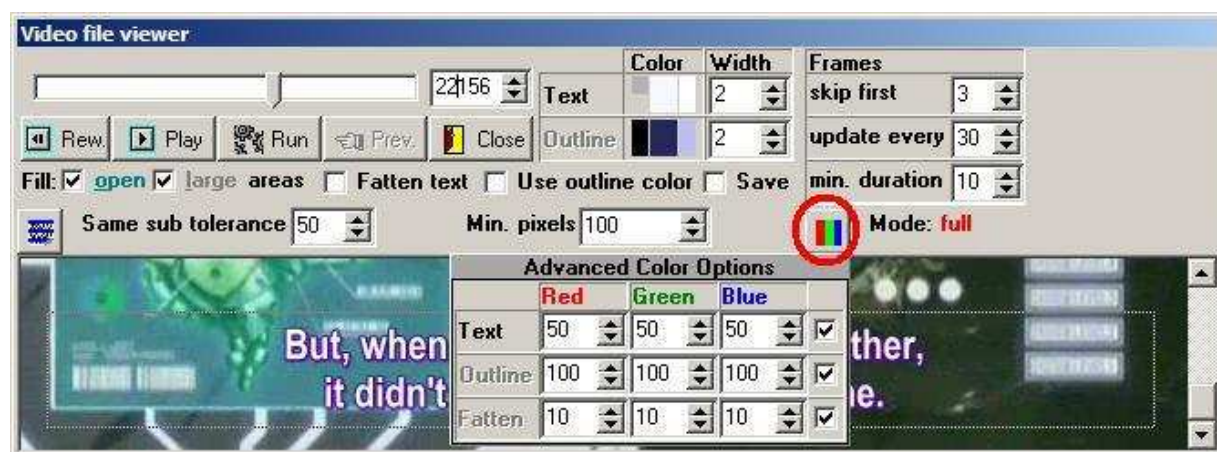


Right-click and **drag** to define a rectangle around the region where the subtitles appear. Make it large enough so that it encloses the subtitles and has enough room on either side for cases when the subtitles occupy a larger area. You can then resize the window to take less space and show only the subtitle region as in the image below. Be careful for cases when there are more lines in the subtitle. You can always stop processing and redefine the region by right-clicking and dragging.



The area encircled in **red** above shows the **Text** and **Outline Colors**. The three colored rectangles in each group show the darkest, detected, and lightest color respectively. Move the cursor inside the rectangle defined earlier (the cursor changes into a cross). **Left-click** **INSIDE** a character (its **white** area) so that SubRip can detect the **Subtitle Color** and **Width**. Look at the area circled in **red** above to confirm: the **detected colors** should match what you see in the **video window**. If not, click again inside another character. Try one that has a vertical line, like "L", "I", "d", "h", etc., and click inside the vertical line. The **Width** boxes should typically show values between 1 and 5 pixels. Anything larger would probably mean that detection was unsuccessful. These values are used for validation, and limit how far around a pixel SubRip searches for neighbors of a similar color. You can also set or change the colors manually by clicking inside the middle (larger) rectangles in the area encircled in **red** above.

If after several tries the detection still does not seem to work, press the button encircled in **red** below to show the **Advanced Color Options** panel. Then, try lowering the **Text Color Tolerance** values (the color of the outline may be too similar to the color of the text). You can change the **Tolerances** for all color channels simultaneously (if the checkboxes in the rightmost column are checked) or for each color channel individually. For example, if the subtitles are **white** and the outlines are **blue**, you may want the **color tolerance** in the **blue** channel to be larger, to compensate for the blurring caused by compression. The **Outline Color** can be used to restrict false guesses: only pixels of the **Text Color** that are close enough to pixels of the **Outline Color** are marked as text. The size of the exploration window is the **Outline Width** value. If the subtitles do not have an outline, simply uncheck the **Use outline color** checkbox and adjust the **Text Width** value manually, after verifying that the text color in the colored rectangle looks correct.



In the main window, a rectangle the size of the selected region will appear, with the subtitles in **white** and the outlines in **red**, as shown below. If the subtitles do not have outlines, fake **red** outlines are added based on proximity to **white** areas. If the subtitles do not show up properly (the lines are too thin, or irregular), try playing with the **Text** and **Outline Widths** or increasing the **Text Color Tolerance** value. Ideally, even on a bright background, you should only see the text in **white** in the main window. If large bright areas also show up as **white**, try checking the **fill open** and **large areas** checkboxes. **Open** areas include areas that touch the border of the rectangle, shown below in **green**. **Large** areas are areas that are taller or wider than a character (10 times the value in the **Text Width** field), shown below in **gray**. Note that the **large areas** on the left are still **white**, because they are not large enough. You can try lowering the **Text Width** value to compensate.



If the subtitles always appear at the same position during the video, press the button encircled in **red** below to show the **Inter-line Options** panel. Check the **Draw lines on top** checkbox. Leave the **Fill sides** checkbox unchecked for now.



Set the **Line Count** to how many lines of text there are in the subtitle. Next, adjust the **Top line offset** so that the top **blue** line just about touches the top of the highest character on the top line. If the **Fill open areas** checkbox is checked, areas that touch the **blue** lines are also considered **open**, and will be filled with **green**, so you need to set the **Top line offset** value so that all characters are still **white**. This helps eliminate false guesses when the background behind the subtitles is **white**. Next, set the line **Height** so that the second **blue** line just about touches the bottom of the lowest character on the first line. Finally, set the **Space** value so that the bottom of the second **blue** line just about touches the highest character of the second text line. The final result should look like the image below. Note that the **large areas** on the left are now **green**, because they are considered **open areas**, since they touch the **blue** line between the subtitles. Also, the **Line Height** value set here will be used in the routine that fills **large areas**.



Finally, you may try checking the **Fill sides** checkbox in the **Inter-line options** panel. Select the **Text Alignment**. This option tells SubRip to start from the **left**, **middle**, or **right**, and fill the areas where it can't find **white** pixels close enough to other **white** text areas with **fuchsia**. The final result should look like the following image:



This particular frame is a very bad case, because of the **white** objects behind the text. The previous image was obtained without using the **Outline Color** as a guide (the **Use outline color** checkbox was not checked). The next image shows what happens in this frame when the **Use outline color** checkbox is checked. Notice that there are **white** areas that are not text.



This problem can sometimes be solved by lowering the **Text Color Tolerance** values, but that may lead to very **thin** or **irregular** characters, as shown below. This is a problem because **thin** characters may become disjoint or may be skipped altogether if the values in **Options -> Advanced OCR Setup -> Character Setup** are small. Also, **irregular** ("eaten by ants") characters will require you to type in the correct text a lot more frequently.



Instead, by leaving the **Text Color Tolerance** values high (>50), and setting the **Text Width** high also (>5), the entire background area is interpreted as text, but becomes large and is filled with **gray**, as in shown the next image. The color does not "bleed" into the letters because of the outline, but that is not always the case. Also, increasing the **Text Width** value significantly slows down processing, because the exploration window is larger, so only use this combination of settings when everything else fails.

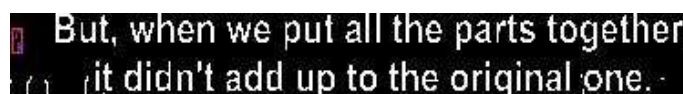


Another way to deal with thin and irregular characters is to use the **Fatten text** feature. The image below shows the result. Notice that the characters are thicker. This also helps reduce the number of times you need to type a character in the **New character(s)** window. The process is controlled by the **Fatten Color Tolerance** values in the **Advanced Color Options** panel. The values for each color channel are relative to the darkest and lightest **Text**

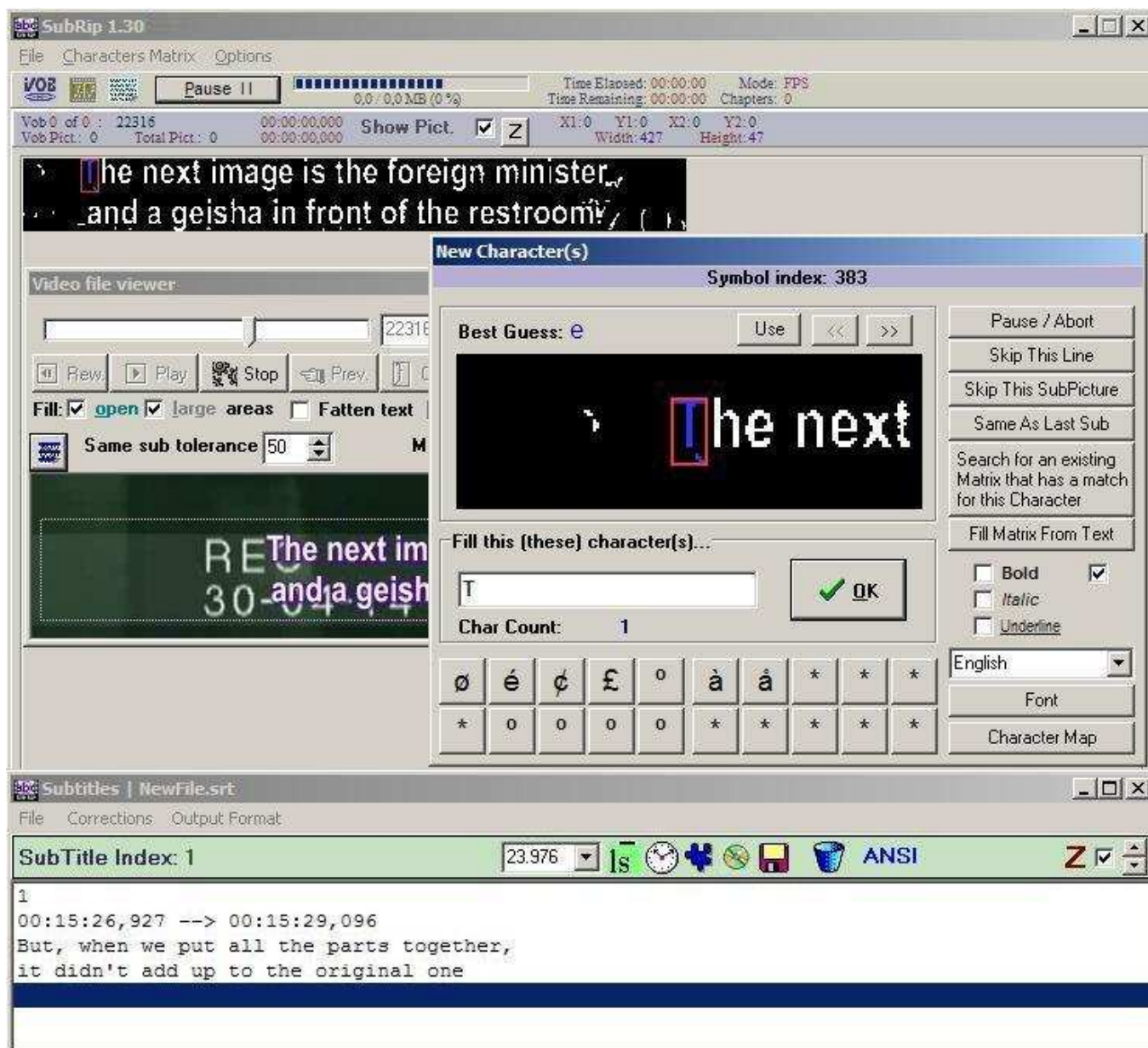
Colors. They control how different a pixel's color can be from the **Text Color** in that channel in order to still be considered for fattening. The darkest and lightest **Fatten Colors** can be seen as colored rectangles right below the darkest and lightest **Text Colors**. If the subtitles have outlines, these tolerances should be larger in the color channels that have larger differences between the **Text Color** and the **Outline Color**.



The purpose of this entire process is to make only the text show up as **white** in the image. Several frames are then accumulated into a black and white image, and the other colors are ignored. The **Same sub tolerance** value tells SubRip by how much the number of detected **white** pixels should vary from frame to frame in order to conclude that the subtitle has changed. The **Min. pixels** value tells SubRip what is the minimum number of **white** pixels that need to be detected to trigger processing. The image that the OCR process is run on looks like the one shown below:



When you are satisfied with the detection parameters, press the **Rew.** button to go to the start of the video, then the **Run** button to start the OCR process. The OCR will be similar to what you see when ripping subtitles from DVDs. You can press **Ctrl+Enter** to fill in the **Best Guess**, then **Enter** to accept it, or press the **Use** button to do both in one step. You can press **Ctrl+Left** and **Ctrl+Right** to grow or shrink the **text selection** (the characters in the **red** selection rectangle) when you encounter **disjoint characters** - for example, when an "O" is split into "(" and ")". Also, just press **Enter** for **white spots** - background areas detected as subtitles such as the one in the **red** selection rectangle in the image above. This way, you are in fact telling SubRip to ignore similar looking **white spots**.



If at any time you see that the subtitles are no longer detected correctly, you may need to change the detection parameters. Press the **Pause/Abort** button in the main window, change them, then press the **Continue** button, just as when processing DVD subtitles. You may also press the **Prev.** button to go the first frame of the last subtitle in the video. This will erase the last subtitle from the text window, and re-run the detection with the new detection parameters.

If the same subtitle shows up more than once, you may continue to fill in the characters to train the OCR (exact duplicates will be detected and joined automatically), or you can press the **Same As Last** button to tell SubRip to go to the next subtitle. If a subtitle is repeated many times, you may need to modify the settings, either by increasing the **Same sub tolerance** value or by tweaking the **Text**, **Outline** and **Fatten Tolerance** values.

If the subtitles appear gradually, set the **Skip first** value to some number greater than 0 to skip that many frames before starting to accumulate frames. After **Min. duration** frames are accumulated, the next frames are just compared with the accumulated image. This speeds up the detection process. The **Update every** value tells SubRip to redo the accumulation process every that many frames. In **accumulate** mode, **white** pixels from different frames are ORed together (added), and in **compare** mode, **white** pixels from different frames are ANDed together (subtracted). Comparison is faster than accumulation, because no other processing is done besides thresholding color values, but may fail to detect when subtitles disappear if the background is entirely **white**. If this situation is encountered often in a video, just set the **Update every** value to **1+Min. duration** to ensure that the **compare** mode is never used. This will slow down the recognition, so only use it if needed, otherwise leave the **Update every** value

at 30 frames or so.

If you check the **Save** checkbox, a back and white bitmap (.pgm) file will be saved for each frame, containing only the characters that were recognized. The areas that were skipped (by pressing **Enter** in the **New character(s)** window for a **NULL** character) are not marked: notice that the **white spots** on the left side of the previous image are no longer present in the image below. The bitmaps, in combination with an index file, can be used later for subtitle removal.

But, when we put all the parts together,
it didn't add up to the original one

Suggestions / questions to this guide?

Name

Message

[?]



Write these letters:

QSS3

Send

[Reklama](#)

Hosting Blueboard.cz

Ke každému hostingu doména za 1 Kč.

Sean Zhou

11.3.2013 (01:32) Reply # X

Would you please add Chinese OCR? seancane@gmail.com

Dennis Lundin

13.1.2013 (21:02) Reply # X

Hi!

I am using Subrip to convert extracted sup files from my mkv files and ocr them then save them to srt. Problem is i always have to enter to many letters manualy. I go crazy on the amount of time it takes sometimes. At first the program seamed to learn and after 2 or 300 it started to run more on auto and only stoped sometimes and i needed to correct. Now i have to sit ther and manually enter letters over and over again. I like this program and i must use it. Is there a way so i dont have to enter that many letters manually ?

G

18.9.2012 (04:57) Reply # X

This is a very Extensive and Great Guide but I am a hard time following it my