@rySubRipper

v1.0

by @rySoft

USER GUIDE

Table of contents

1 Foreword

For several years I tried to find on the net the subtitles for some rare tv shows that came with hardcoded subtitles in their original language.

Having tried the few available applications available for ripping subs without success. I've finally started this project for personal use.

My objective was to quickly have a working application that produces accurate subtitles for series using the same set of characters through all the episodes. Of course, you can use @rySubRipper to rip movie subtitles. Just be aware that by starting each time with a different set of subtitle characters, the ripping process will take longer because of the OCR learning curve.

With that goal in mind, I chose a "binary comparison" technology as opposed to a standard OCR (more flexible but also more complex and unreliable).

That said, accuracy has its price and ripping a tv show episode takes a lot of user time even once the machine learned the basic characters.

If you are looking for an application where you press a button and leave for your computer to work alone, you can stop reading now and delete it from your disk.

The quality of the video and the contrast between the subtitles and the background images will have a huge impact on the ripping time and the manual work. You can have scenes where the machine turns alone for several minutes and scenes where you have to interact every second. Ripping a 20-minute episode can take a few working hours.

As the first user, I tried to simplify the user task as much as possible by adding new features I identified while using the app.

If you are still there and willing to give the app a try, let's see how it works.

2 Understanding the application

The logic behind the application is simple:

Inside a frame:

1) You identify a character on the screen.
2) You try to find the best match for the character in the characters database.

Between frames:

1) Identifying different frames
2) Identifying empty frames

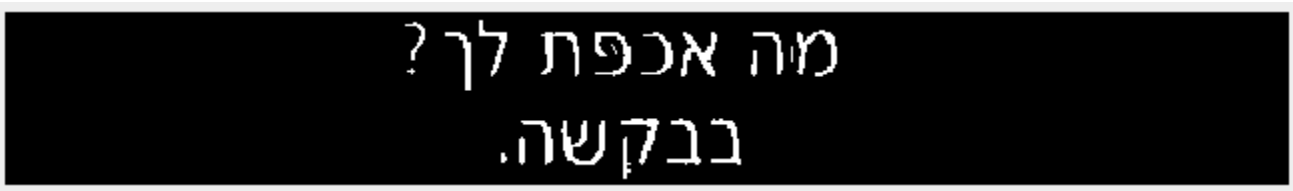
2.1 Identifying a character

Subtitles are normally positioned in the same place, have the same color, are centered on the screen, and have one or two lines (it is a very unusual case to have three lines so I didn't add a third line because it would add a lot of processing time for nothing. If this case appear you will have to treat it manually).

So, the first thing is to define exactly where the lines are. The most accurate the top and bottom of each line are set, the better the application will work. As you can see below, the ***"capture box"*** (blue lines) match exactly the top and bottom of the higher and lower characters. Use the ***"set capture box"*** button together with the coordinates ***"L1 (x,y,w,h) and L2 (y)"***. Subtitles incorrectly centered on the "capture box lines" might not be detected by the application.

The second set will be to pick the right "ink" color for the characters. You test the best configuration by playing with the colors and the ink tolerance and testing the results with the ***"capture box"*** button. The idea is to minimize the tolerance (to avoid as much as possible the background interference) while still capturing the text correctly.

You should see something like that:



The captured box will display the ink color in white and the rest in black. The background will be gone.

The next step will be to identify each individual character. As we have now two different lines, the only thing to do will be to identify full vertical black lines separating the white characters and scan the line left to right or right to left (depending on setting of the button ***"L→R"*** or ***"L←R"*** or the variable ***"read_dir"***).

As far as the background color doesn't overlap with the ink one, the auto mode will work very well with very few manual interventions needed.

When the background matched the ink color, manual interventions will be required and, in some cases, you'll need to manually enter the full subtitle.

You'll need to decide for these parts, depending on how bad the background is, if you run auto or you work in manual mode.

In the following example there is no black vertical line so the app will consider that each line is only one character.

2.2 Finding the best match

Once a character has been isolated on the image (all the black borders removed) we'll have to compare it with similar characters stored in the application database.

A character has several attributes: height, width, a distribution of black&white pixels and a vertical position on the line. The comparison will only be performed with characters of the same size so, in the following example, the next two characters will not be compared together as they have a different height.

For character of the same size, a matching coefficient (0 will be completely different, 1 will be an exact copy) will be calculated based on the black&white pixels positions and the vertical position on the line. A specific parameter (***vert_pix_match_coef***) will allow you to define how to the vertical position affects the black&white pixels coefficient.

Once the application have found the best match, it will decide if it automatically uses it or if it needs your input based on the ***"character tolerance"***.

There's a global tolerance definition ("OCR tolerance" on the application window, "char_tol" on the global parameters) to define when the application accepts or not the character.

For example, 0.97 means that we need a minimum match of 97% to accept the character.

When a new character is detected and you enter its value manually, the application will automatically affect to it the global tolerance value.

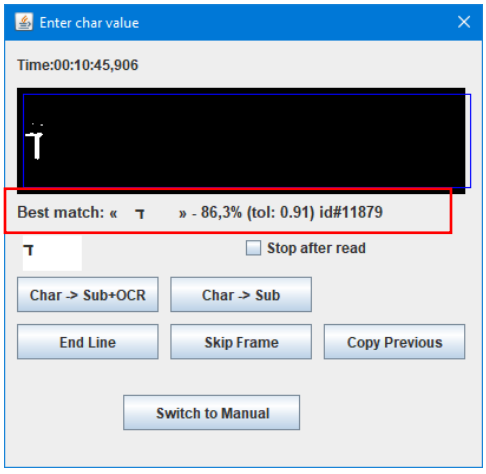
You can tune the matching tolerance for each character with the **"Edit character tolerance"** button. I recommend you to start with a high value (i.e. .95 or .96) and then fine tune easy to recognize characters.

You can activate the OCR stats to see the best matching values of each non recognized characters so see which characters are similar and which ones not.

In my tests, I've managed to reach a perfect accuracy having some chars at 0.89, while others requested .92, .95 or even .97 (small symbols like dots, commas or quotes that can easily match spots).

2.2.1 The new char value pop up window

When there's no match reaching the tolerance, the pop up window "enter a new char value" will show you the best match found and the tolerance defined for that character (or null if no character in the database matches the new character dimensions).



The parameter ***"char_prop_tol"*** (character proposal tolerance) will allow you to define when a best match not reaching the character tolerance will be proposed for data entry.

In the previous example, the character tolerance is .91. If we define a ***"char_prop_tol"*** of .15, only best matches between .76 and .91 will be proposed.

The possible actions on the pop-up window are:

- **Char -> Sub+OCR:** this button will add your character (can be empty, one character or several ones) both to the current subtitle and into the OCR database for its "learning". It is not recommended to store more than two-character strings or strings with spaces. If an entry corresponds to an empty string, don't put a space on it, the OCR algorithm will calculate if it needs a space or not based on the ***"min_spc_pix"*** parameter.
- **Char -> Sub:** this button will add your character only to the current subtitle but not to the OCR database. It should be used mostly for bad background issues: characters that could be easily confused with others, multiple characters, multiple characters with spaces (you need to put the space for the sub to be right), white spots will be hardly matched in future frames, big white spots (because a big matching percentage could hide some characters in future matches).
- **End line:** this button is to be used when there is nothing else on the line but white spots (will not consider the current character entry).
- **Skip frame:** this button will disregard the current frame (containing only white spots) and search for the next one.
- **Copy previous:** this button will copy the text from the previous stored frame and search for the next one. To be used in cases where the same subtitle applies but the application detected a different frame (see the chapter "*detecting different frames"*).
- **Switch to manual:** will stop the reading of the subtitle on the current frame and switch to "manual" mode.
- **Stop after read:** this option will continue reading the current subtitle to the end and then switch to "manual" mode. Be aware that you will have to uncheck the box either on the pop up or on the OCR reading pane to come back to the "auto" mode.

2.3 Identifying different frames

When running in auto mode (or clicking on the button *"Seek next sub"*) the application will jump to the next frame based on the parameter ***"step_frame"***. After a lot of use, I've found that 500ms is a good compromise between processing speed, subtitle detection and subtitle time accuracy (using the ***"sub_adjustment"*** parameter to compensate in some degree this jump) but you are free to try different settings.

When a new frame that is visited, the application will compare it with the previous one to see if there's a difference (either an empty frame or a change of subtitle).

The current algorithm compares two different characteristics:

1) Number of white points: if the number of white points is different then we consider that the current frame is different from the previous one. Two parameters permit to play with this comparison ***"frame_tol_pct"*** and ***"frame_tol_pix"***. The first one will determine the maximum allowed difference in percentage, the second one the maximum different in pixels. If one of them is reached, the application will consider that the frame has changed.
2) Text line samples: as a subtitle is centered on the line both vertically and horizontally, we can take a sample of each line (a line of a pixel height across the text) to compare it between frames. If the sample line is not similar, then we can consider that the frame has changed. This helps to identify both frames

with similar number of whites but different text and frames with different number of whites (i.e. spots on the corners) but identical text.

2.4 Identifying empty frames

To detect empty frames, we also use the two above mentioned characteristics.

1) Number of white points: if the number of white points is less than ***"min_frame_pix"***, we consider that the frame is empty.
2) Text line samples: If the samples in both text lines are empty, we consider that the frame is empty.

3 Getting started

@rySubRipper uses VLC to read videos, so you need to have it installed. If you don't have it installed, you can download it at https://www.videolan.org/

Important: @rySubRipper is a java application (runnable jar) so it is "machine independent". However, the VLC plugin has some constraints. If you are in a 64bit environment, you need a 64bit version of VLC installed or it will not work.

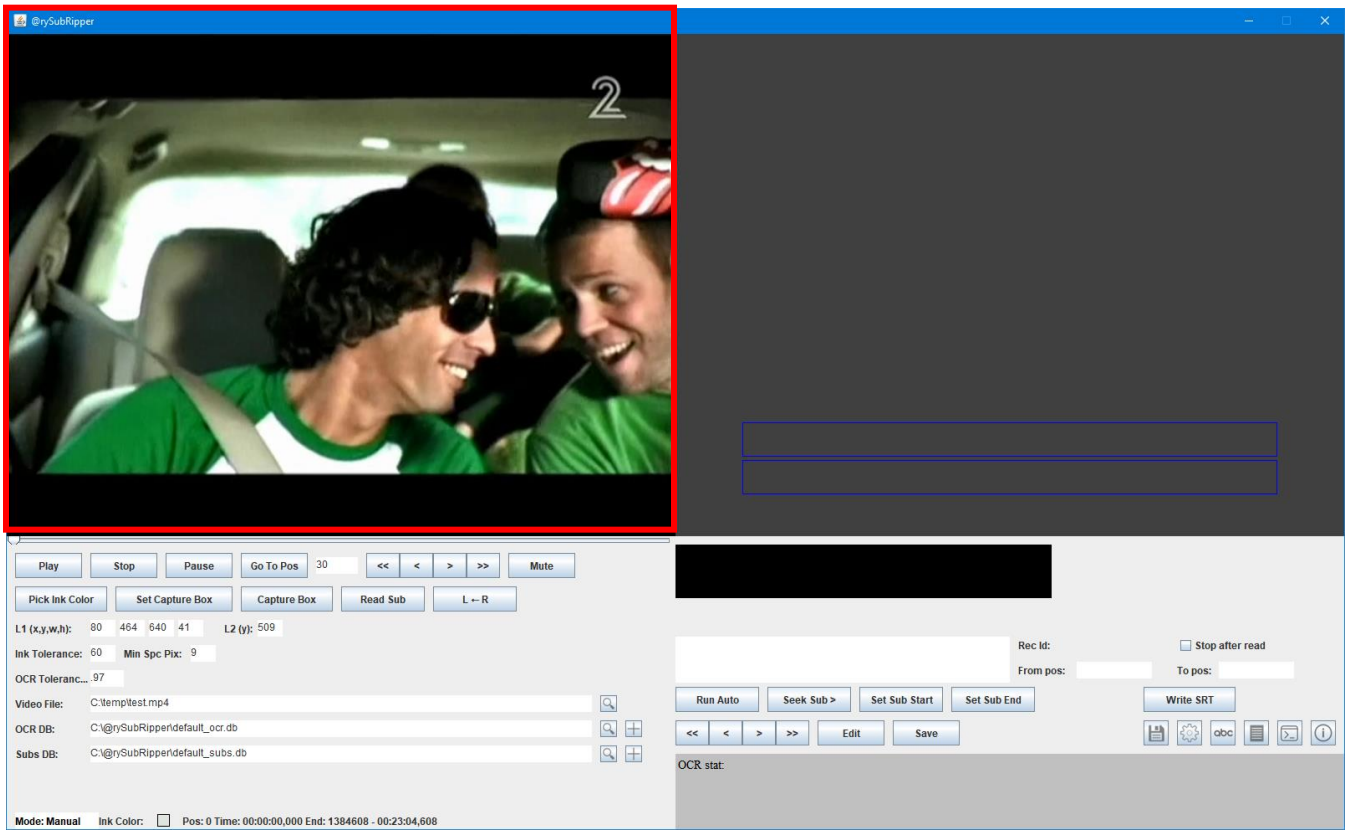
In addition, in some cases, the application will automatically detect VLC but you will probably need to edit the ***config.ini*** file and set the VLC path there (by default is set up to C:\Program Files\VLC):

[Path]
vlc = C:\Program Files\VLC

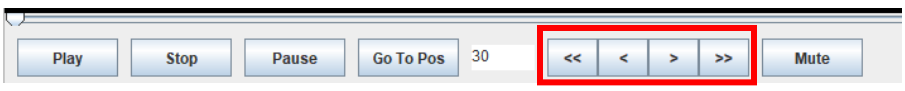
The application is a .jar executable file so you also need to have Java installed in your computer.

@rySubRipper comes with some default databases and parameters. You can start with that and play with the parameters once you are familiar with the app.

Once this set up, you can just load a video file and see if it is correctly loaded in the player panel on the left side of the application.

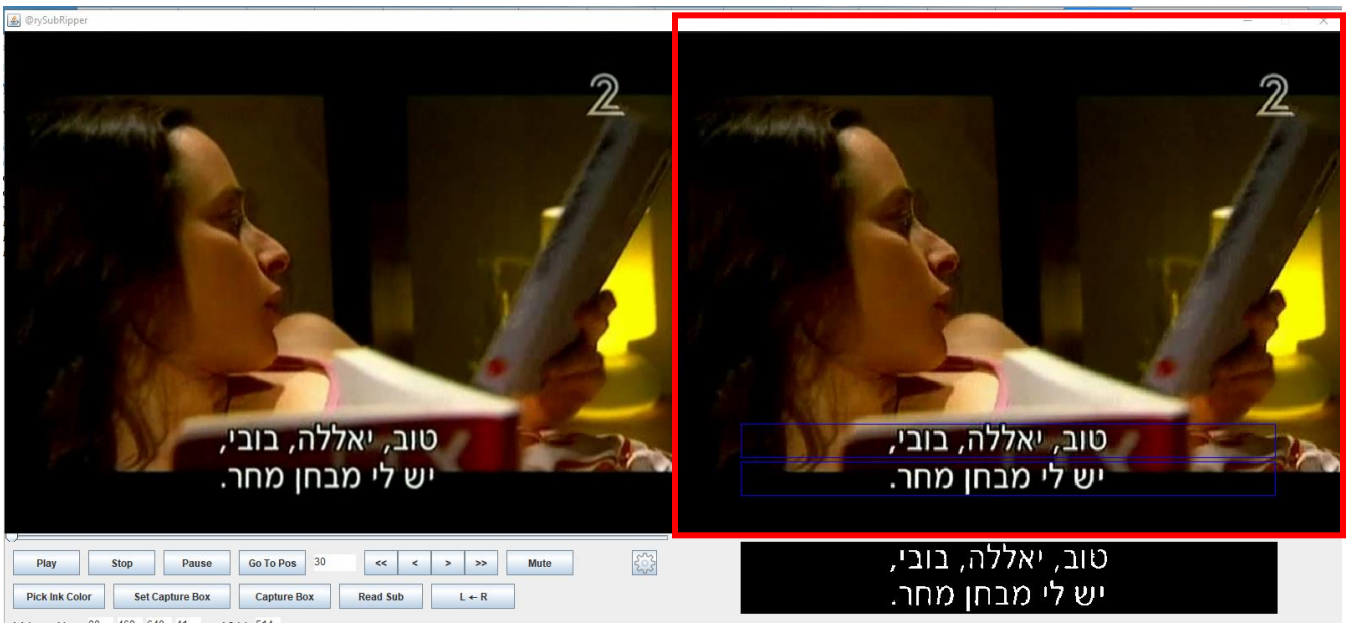


The video should be paused in the first frame and you need to keep it paused while working. You can play with the player controls advance/rewind to see if everything is working.



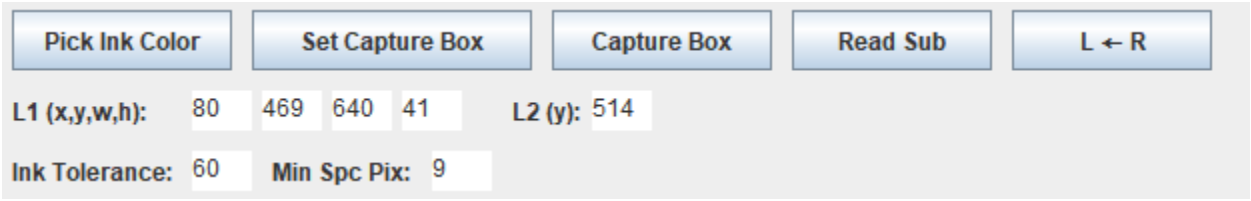
If that's the case, you are ready to go.

Move to the first available subtitle with a double line and click on the player image or on the **"capture box"** button to copy the player frame onto the OCR image (on the right side of the screen).



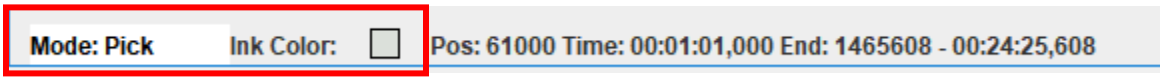
Now you have to set up the capture box lines to match the subtitle lines as explained on the chapter "2.1 – Identifying a character".

You will need to play with the box coordinates "L1 (x,y,w,h) and L2(y)" and click the "set capture box" to check the results.



Once the lines are correctly set up, you have to click on the "Pick ink color" button and then click on the OCR image to select the subtitles ink color.

Check on the status bar that you are in Pick mode. The select color will also appear there.



You can test the picked color with the ***"capture box"*** button and see the results on the ***"captured box"***.

You can play with picking colors and the ***"Ink Tolerance***" value until you get the right combination.

With that set up, you can start ripping the subtitles.

4 Ripping a subtitle

First step when you are ripping a subtitle is to define the different databases (subtitles, ocr) and selecting a video file to rip.

There are default databases for the OCR (stores OCR characters information and parameters) and for the subtitles (stores the ripped subtitles).

I recommend having a different OCR database for each tv show if they have different subtitle characters.

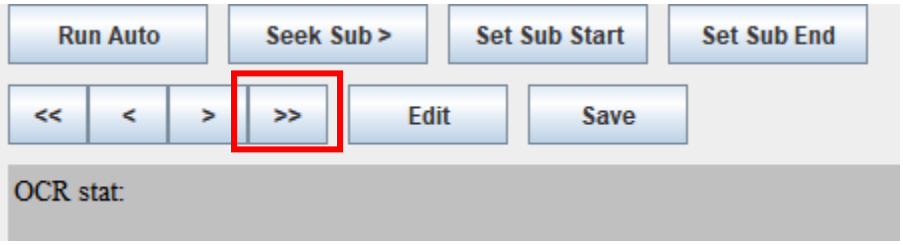
A bigger OCR database will mean more processing time and even mismatched characters with no visible benefit.

Subtitles are stored by video file name so there are no constraints there, but my personal preference is to have a different base per episode.

As subtitles are stored in the database you don't need to do the full video all in a row. You can stop and restart the process as many times as you need.

The first time you start with a video, would like to position in the first subtitle frame (using the player navigation buttons) before starting the ripping (you can start from the first video position, but you will spend time for nothing if the first subtitle appears a few minutes after).

If you want to continue with a video that you left in the middle, after loading a video you can navigate to the last recorded sub with the "go to the last subtitle button" on the OCR toolbar



and then position on the next subtitle to rip with the player navigation controls.

Now you can start the ripping either manually or automatically. You can switch between both methods as frequently as you need it depending on your needs.

4.1 Automatic ripping mode

You just click the ***"Run Auto"*** button and the automatic ripping process will start from the current player position.

The process will continue until the end of the video or until you choose ***"Stop after read"*** or ***"Switch to manual"*** on the ***"OCR character entry pop-up window."***

4.2 Manual ripping mode

The manual mode allows you to manage manually start, end and text.

First step is to position yourself in the first frame of the subtitle to be ripped either with the ***"Seek sub >"*** button on the OCR toolbar or with the player navigation options.

You have several options here:

1) Enter the subtitle manually on the OCR ***"subtitle edition"*** text box.
2) Launch a subtitle reading by clicking ***"read sub"*** button.
3) Click on the ***"Set Sub Start"*** button.

Clicking the ***"Set Sub Start"*** button will set up the starting subtitle position in the ***"From Pos"*** box. If the ***"subtitle edition"*** text box is empty, a subtitle reading will be started, if the box has already some text, only the starting position will be set.

In any case, after the start position is set, you can always edit the subtitle.

Once the subtitle text and start position are set, you have to navigate to the last frame of the subtitle and click on the ***"Set Sub End"*** button to store the subtitle on the database.

You then need to navigate to the start of the next subtitle and repeat the operation or start an auto run.

5 Generating the SRT file

Once you've finished ripping all the subtitles on the video, all the subtitles are stored in the database.

It's time to finally generate your SRT file.

By clicking the ***"Write SRT"*** button you will start the subtitle generation.

The process will first launch several cleaning processes:

- Inverse consecutive numbers (in the case of right to left languages)
- Clean subtitles (remove subtitles with only one special character like dots, commas, quotes…)
- Remove double subtitles (if consecutive subtitles have the same text – this happens when a frame change is detected but the text remains the same-, then it will merge them into only one subtitle)
- Fix subtitle overlaps: in case some manual subtitles treatment overlapped the end of a subtitle with the start of the following one.

Then it will generate the SRT file.

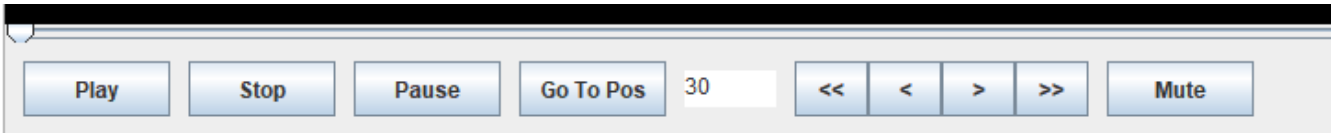
6 Full list of controls

6.1 The player toolbar

These controls don't need too much explanation.

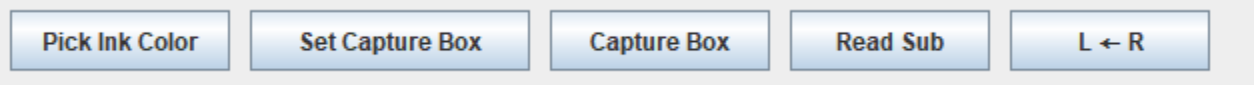
You can navigate through the video file with the scrollbar, by playing the video, jumping to a specific position (in milliseconds) or by the fast and slow back/forward step buttons.

The step of the fast/slow step buttons can be set through the advanced parameters.



6.2 The OCR tuning toolbar

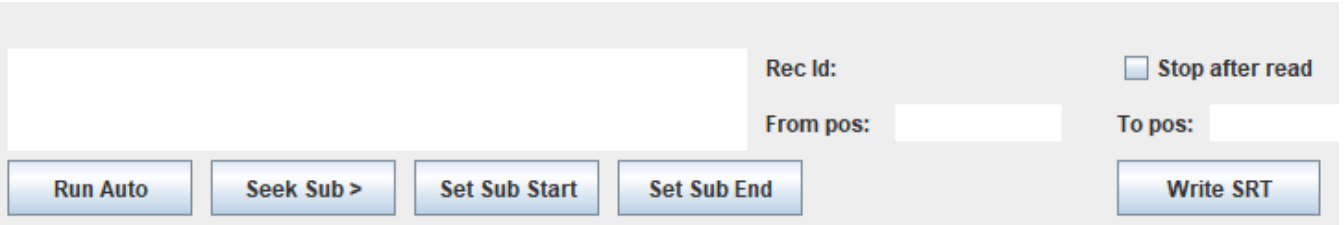
This toolbar of 5 buttons allows you to set up, tune and test the OCR together with the parameters panel below it.



- **Pick Ink Color:** toggles on/off the Pick mode to choose the subtitle color by clicking on the OCR image. The selected color will be displayed on the status bar at the bottom of the application.
- **Set Capture Box:** sets (and displays) the subtitle lines capture boxes on the OCR image based on the ***"L1 and L2 coordinates"*** defined on the parameters panel.
- **Capture Box:** allows you to test the combination of ***"Ink color"*** and ***"Ink tolerance".*** When clicked it will copy a snapshot of the current player frame into the OCR image and then try to capture de subtitles in black&white into the ***"captured box"***.
- **Read sub:** launches a manual OCR reading of the current **"captured box".**
- **L⟶R:** changes the reading direction of the OCR (left to right or right to left)

6.3 The subtitle ripping toolbar

This toolbar allows you to execute the subtitle ripping.



- **Subtitle text box:** shows the current subtitle reading in auto mode. It also allows you to edit the subtitle text in manual mode or in edit mode.
- **From pos:** normally is only used for display but you can set it manually or modify it in manual mode or subtitle edition mode.
- **To pos:** also used only for display. It can be modified manually in edition mode.
- **Stop after read:** when checked, it will automatically switch to manual mode after the ending the reading of the current subtitle.

- **Run auto:** launches the automatic ripping of the video from the current position.
- **Seek sub >:** launches a search for the next subtitle.
- **Set Sub Start:** sets the current position as start position. If the text box is empty, then it launches a reading of the current **"captured box".** If ***"seek after manual"*** parameter is set to 1, launches a Seek Next Sub after the end of the reading.
- **Set Sub End:** in manual mode, it stores a new subtitle into the database by using the "subtitle text box", and "From Pos" values as well as the current player position. In edition mode, saves the current edited subtitle with the current player position as end position.
- **Write SRT:** writes de SRT for the current video file (see chapter 5 – Generating the SRT file).

6.4 The subtitle edition toolbar

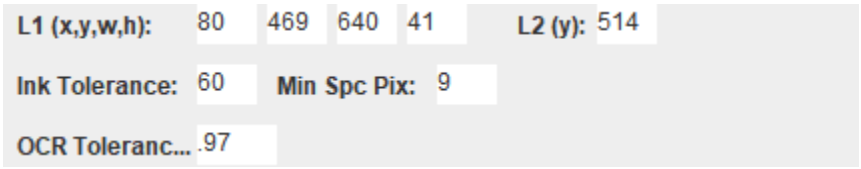
This toolbar allows you to edit ripped subtitles.

- **<<:** moves to the first stored subtitle.
- **<:** moves to the previous stored subtitle.
- **>:** moves to the next stored subtitle.
- **>>:** moves to the last stored subtitle.
- **Edit:** toggles on/off the edition mode. When in edition mode, you are able to edit the text, start and end of a subtitle on the "subtitle text", "pos from" and "pos to" boxes.
- **Save:** when in edition mode, saves to the database the current subtitle with the "subtitle text", "pos from" and "from to" boxes values.

6.5 The additional controls toolbar

This toolbar contains buttons opening different pop-up windows.

- **Save parameters:** saves the parameters of the settings panel into the current OCR database.



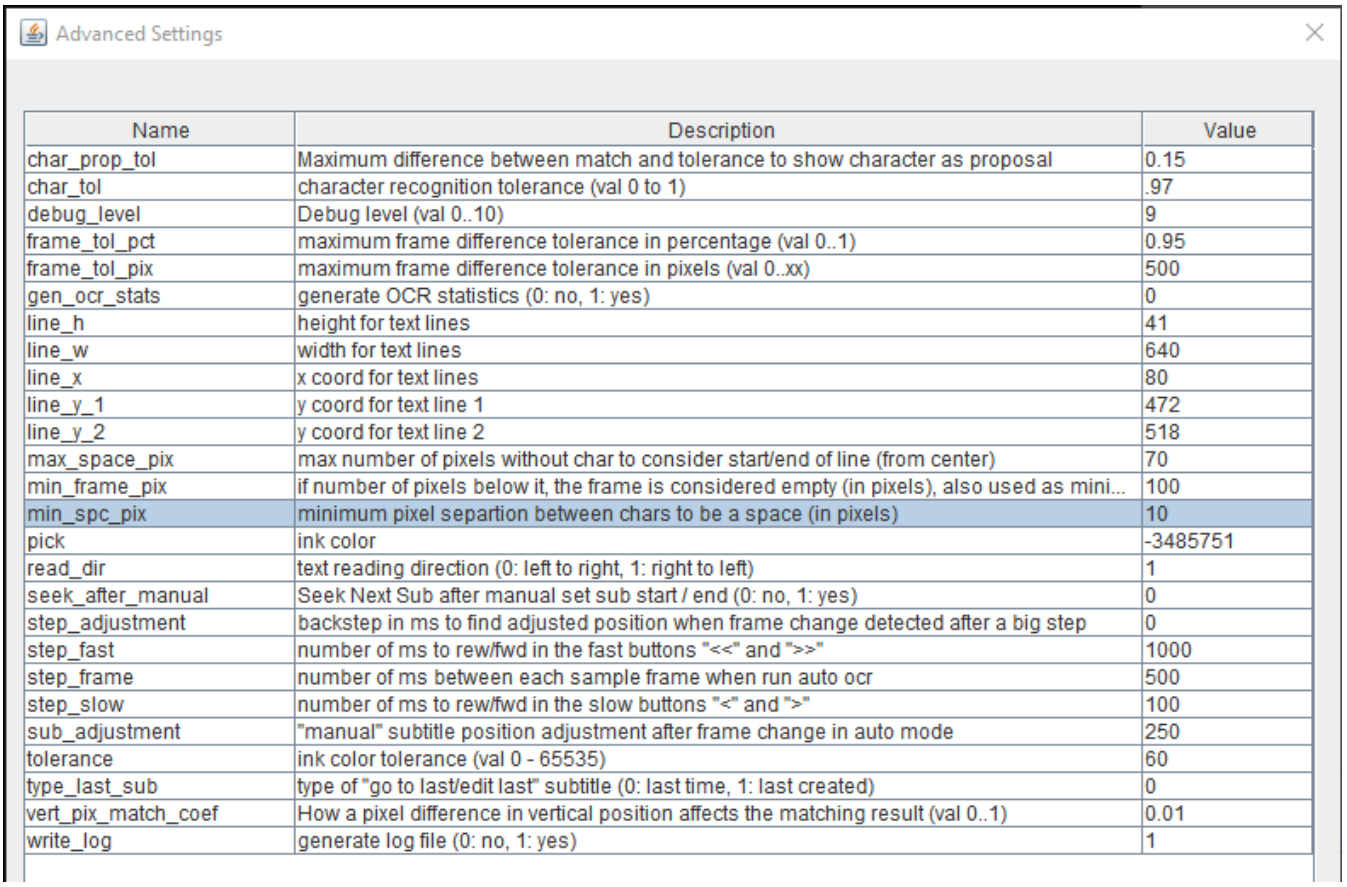
the settings panel

- **Advanced parameters:** opens the advanced parameters edition window (see chapter 7 – Full list of parameters).
- **Character tolerance:** opens the character tolerance edition window.
- **OCR statistics:** opens the character tolerance edition window.
- **Console:** opens a debug console window.
- **About:** opens the about window.

7 Full list of parameters

The advanced settings window allows you to manually edit all the available parameters. The save button will save the values into the database.

All the parameters are described in the window.



Name	Description	Value
char_prop_tol	Maximum difference between match and tolerance to show character as proposal	0.15
char_tol	character recognition tolerance (val 0 to 1)	.97
debug_level	Debug level (val 0..10)	9
frame_tol_pct	maximum frame difference tolerance in percentage (val 0..1)	0.95
frame_tol_pix	maximum frame difference tolerance in pixels (val 0..xx)	500
gen_ocr_stats	generate OCR statistics (0: no, 1: yes)	0
line_h	height for text lines	41
line_w	width for text lines	640
line_x	x coord for text lines	80
line_y_1	y coord for text line 1	472
line_y_2	y coord for text line 2	518
max_space_pix	max number of pixels without char to consider start/end of line (from center)	70
min_frame_pix	if number of pixels below it, the frame is considered empty (in pixels), also used as mini...	100
min_spc_pix	minimum pixel separtion between chars to be a space (in pixels)	10
pick	ink color	-3485751
read_dir	text reading direction (0: left to right, 1: right to left)	1
seek_after_manual	Seek Next Sub after manual set sub start / end (0: no, 1: yes)	0
step_adjustment	backstep in ms to find adjusted position when frame change detected after a big step	0
step_fast	number of ms to rew/fwd in the fast buttons "<<" and ">>"	1000
step_frame	number of ms between each sample frame when run auto ocr	500
step_slow	number of ms to rew/fwd in the slow buttons "<" and ">"	100
sub_adjustment	"manual" subtitle position adjustment after frame change in auto mode	250
tolerance	ink color tolerance (val 0 - 65535)	60
type_last_sub	type of "go to last/edit last" subtitle (0: last time, 1: last created)	0
vert_pix_match_coef	How a pixel difference in vertical position affects the matching result (val 0..1)	0.01
write_log	generate log file (0: no, 1: yes)	1

Equivalences:

Advanced settings	Settings panel
line_h, line_w, line_x, line_y_1, line_y_2	L1 (x,y,w,h) L2(y)
tolerance	Ink tolerance
min_spc_pix	Min Spc Pix
char_tol	OCR Tolerance
read_dir	L⟶R button

Logging level values:

0 - no logging
1 - errors
3 - very important info
4 - important info
5 - process level info
6 - very basic debug info
7 - basic debug info
9 - additional debug info
10 - total debug info