Общие требования ко всем приложениям

1. Записано видео об использовании приложения (с двух ракурсов или совмещенное). Видео должно быть загружено на https://www.youtube.com/. Продолжительность >= 2 минут.   Примеры:

a.   <https://www.youtube.com/watch?v=DYzOSCX6gp0>

b.   <https://www.youtube.com/watch?v=r42z259-HHE>

2. Пользовательская документация доспуна в pdf или html формате и включает в себя:

a.   Как получить и установить приложение

 b.  Как сконфигурировать и запустить приложение

c.   Описание функциональности приложения

d.   Примеры использования

e.   Известные проблемы

3. Подготовлена презентация (ppt) о разработанном демо. Презентация должна содержать следующие части:   
a.    О проекте (описане, функционал и т.д.)

b.           О разработке (ссылки на проектные ресурсы, используемые инструменты, трюки и ловушки, и т.д.).

4. Все материалы проекта (комментарии в коде, документация, демо и пр.) на Английском языке.

5.            Приложение не должно содержать критических ошибок.

**Emotion tracker**

The main aim of this project is to develop a library and demo applications to track and write users emotions (sadness, smile, laugh, etc.) during watching movie. Test recognition of emotions with different conditions. Implement logic to “merge” files with emotions and provide average rating. Implement logic to gather eyes track and to build heat map for user’s attention. All of that should be done by using Intel RSSDK tools.

**Download Sources and Documentation**

**BitBucket**

The URL of the repository is <https://bitbucket.org/valber/emotracker> . One can download it from <https://bitbucket.org/valber/emotracker/get/1f57a3708e1d.zip>

or clone using git:

$ git clone https://bitbucket.org/valber/emotracker.git

**Project tree**

.

├── [**emotracker**](https://bitbucket.org/valber/emotracker/src/308cf96a5df68f8a46173bd04c1dfc35d6baf8b6/emotracker/?at=master) # emotions writer prototype (used only for research │ # purposes)

├── [**library**](https://bitbucket.org/valber/emotracker/src/308cf96a5df68f8a46173bd04c1dfc35d6baf8b6/library/?at=master) # EmoTracker libraries

│ ├── [**CSharpLibrary**](https://bitbucket.org/valber/emotracker/src/308cf96a5df68f8a46173bd04c1dfc35d6baf8b6/library/CSharpLibrary/?at=master) # C# wrapper for native library

│ │ ├──[Build.docx](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/library/CSharpLibrary/Build.docx?at=master) # Build instructions

│ │ └──[CSharpLibrary.sln](https://bitbucket.org/valber/emotracker/src/308cf96a5df68f8a46173bd04c1dfc35d6baf8b6/library/CSharpLibrary/CSharpLibrary.sln?at=master) # VS 2015 solution

│ ├── [**emotracker**](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/library/emotracker/?at=master) # Native C++ Windows library

│ │ ├──[**docs**](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/library/emotracker/docs/?at=master) # API documentation

│ │ ├──[Build.docx](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/library/emotracker/Build.docx?at=master) # Build instructions

│ │ ├──[emotracker.sln](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/library/emotracker/emotracker.sln?at=master) # VS 2015 solution

│ │ └── ... # etc.

│ └── ... # etc.

├── [**rssdk2video**](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/rssdk2video/?at=master) # Utility for converting rssdk format to video formats

│ │ # (it should be in utils dir)

│ ├── [Build.docx](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/rssdk2video/Build.docx?at=master) # Build instructions

│ ├── [rssdk2video.sln](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/rssdk2video/rssdk2video.sln?at=master) # VS 2015 solution

│ └── ... # etc.

├── [**samples**](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/samples/?at=master) # Examples for using

│ ├── [**EmoMerge**](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/samples/EmoMerge/?at=master) # C# application to merge emotions TTML files

│ │ └──[EmoMerge.sln](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/samples/EmoMerge/EmoMerge.sln?at=master) # VS 2015 solution

│ ├── [**EmoTracker**](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/samples/EmoTracker/?at=master) # C# application used EmoTracker library for emotions

│ │ │ # recording

│ │ ├──[Build.docx](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/samples/EmoTracker/Build.docx?at=master) # Build instructions

│ │ ├──[EmoTracker.sln](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/samples/EmoTracker/EmoTracker.sln?at=master) # VS 2015 solution

│ │ └── ... # etc.

│ └── ... # etc.

├── [**utils**](https://bitbucket.org/valber/emotracker/src/84e060bcad8f5a7200ab09518421b7f61bb4c614/utils/?at=master) # Utilities to use emotions tracks

│ ├── [**GazeHeatMap**](https://bitbucket.org/valber/emotracker/src/1f57a3708e1d07eb6d19576a40f15351ad977845/utils/GazeHeatMap/?at=master) # Map recorded gaze onto the video

│ ├── [**GazePainter**](https://bitbucket.org/valber/emotracker/src/b99019e0cb5b052de0c93cee63d8bfcb2136b020/utils/GazePainter/?at=master) # Map recorded gaze onto the video

│ └── ... # etc.

├── [Doxyfile](https://bitbucket.org/valber/emotracker/src/b99019e0cb5b052de0c93cee63d8bfcb2136b020/Doxyfile?at=master) # Doxygen configuration file to build documentation

├── [survey.doc](https://bitbucket.org/valber/emotracker/src/b99019e0cb5b052de0c93cee63d8bfcb2136b020/survey.doc?at=master) # About this project

├── [survey.pdf](https://bitbucket.org/valber/emotracker/src/b99019e0cb5b052de0c93cee63d8bfcb2136b020/survey.pdf?at=master) # About this project

├── [test.ttml](https://bitbucket.org/valber/emotracker/src/b99019e0cb5b052de0c93cee63d8bfcb2136b020/test.ttml?at=master) # TTML emotions record sample

└── ...

**Installation**

**Prerequisites**

emotracker:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C++ 2015**

**Intel® RealSense™ SDK 2016 К2**

library/emotracker:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C++ 2015**

**Intel® RealSense™ SDK 2016 К2**

library/CSharpLibrary:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C# 2015**

**Intel® RealSense™ SDK 2016 К2**

**emotracker library**

rssdk2video:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C++ 2015**

**Intel® RealSense™ SDK 2016 К2**

**OpenCV2**

samples/EmoMerge:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C# 2015**

samples/EmoTracker:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C# 2015**

**Intel® RealSense™ SDK 2016 К2**

**emotracker library**

**CSharpLibrary library**

utils/GazePainter:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C+ 2015**

utils/GazeHeatMap:

To build it you need:

**Microsoft Visual Studio 2015**

**Microsoft Visual C# 2015**

**Installing**

library/emotracker:

1. Open emotracker.sln with Microsoft Visual Studio 2015
2. Check path to RSSDK include directory
   1. Project -> Properties
   2. C/C++ -> General -> Additional include path it should contain RSSDK include path: $(RSSDK\_DIR)/include
3. Check path to RSSDK libraries directory
   1. Project -> Properties
   2. Linker -> General -> Additional library path it should contain RSSDK library path $(RSSDK\_DIR)/lib/$(PlatformName)
4. Build and save the library emotracker.dll, under the local bin directory

library/CSharpLibrary:

1. Open CSharpLibrary.sln with Microsoft Visual Studio 2015
2. Open the Solution Explorer (View -> Solution Explorer), expand Solution -> CSharpLibrary -> References and check if there is reference to the libpxcclr.cs library
   * If not, then right click on the Reference -> Add reference… to open Reference manager
   * Click Browse tab, and then Browse… button to find location of libpxcclr.cs.dll in your file system
3. Build and save the library CSharpLibrary.dll, under the local bin directory

samples/EmoTracker:

1. Open EmoTracker.sln with Microsoft Visual Studio 2015
2. Open the Solution Explorer (View -> Solution Explorer), expand Solution -> EmoTracker -> References and check if there is reference to the libpxcclr.cs library
   1. If not, then right click on the Reference -> Add reference… to open Reference manager
   2. Click Browse tab, and then Browse… button to find location of libpxcclr.cs.dll in your file system
3. Open the Solution Explorer (View -> Solution Explorer), expand Solution -> CSharpLibrary -> References and check if there is reference to the CSharpLibrary library
   1. If not, then right click on the Reference -> Add reference… to open Reference manager
4. Click Browse tab, and then Browse… button to find location of CSharpLibrary.dll in your file system
5. Build and save the EmoTracker.exe application, under the local bin directory

rssdk2video:

1. Open rssdk2video.sln with Microsoft Visual Studio 2015
2. Check path to RSSDK and OpenCV include directory
   1. Project -> Properties
   2. C/C++ -> General -> Additional include path it should contain RSSDK and OpenCV include path, t.ex. : $(RSSDK\_DIR)/include; $(OPENCV\_DIR)/include;
3. Verify also if Macros $(OPENCV\_DIR) points to correct OpenCV path
   1. To display **Property Manager**, on the menu bar, choose **View**, **Other Windows**, **Property Manager**.
   2. Expand rssdk2video -> Debug | Win32, right click on PropertySheet to open **Property Page** dialog
   3. Select **User Macros** tab, and verify correctness of OPENCV\_DIR macros defines the path to OpenCV library location
4. Check path to RSSDK libraries directory
   1. Project -> Properties
   2. Linker -> General -> Additional library path it should contain RSSDK and OpenCV library path, t.ex.: $(OPENCV\_DIR)/$(PlatformTarget)/vc12/lib; $(RSSDK\_DIR)/lib/$(PlatformName)
5. Build and save the rssdk2video.exe utility, under the local bin directory

samples/EmoMerge:

1. Open EmoMerge.sln with Microsoft Visual Studio 2015
2. Build and save the EmoMerge.exe application, under the local bin directory

utils/GazePainter:

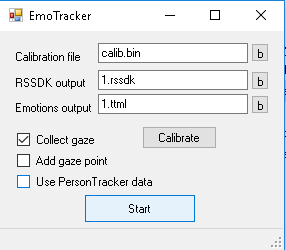
1. Open GazePainter.sln with Microsoft Visual Studio 2015
2. Build and save the library GazePainter.exe, under the local bin directory

samples/GazeHeatMap:

1. Open GazeHeatMap.sln with Microsoft Visual Studio 2015
2. Build and save the library GazeHeatMap.exe, under the local bin directory

**Usage**

**Case 1. Track and write users emotions during video playback**

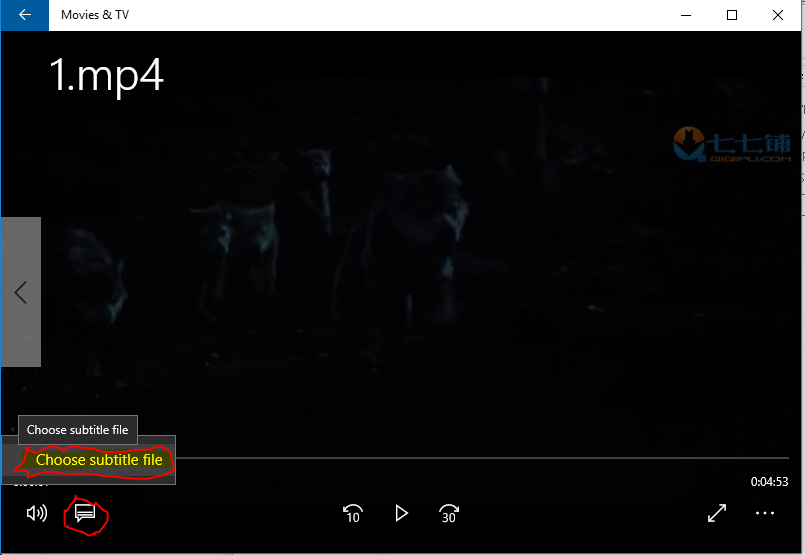
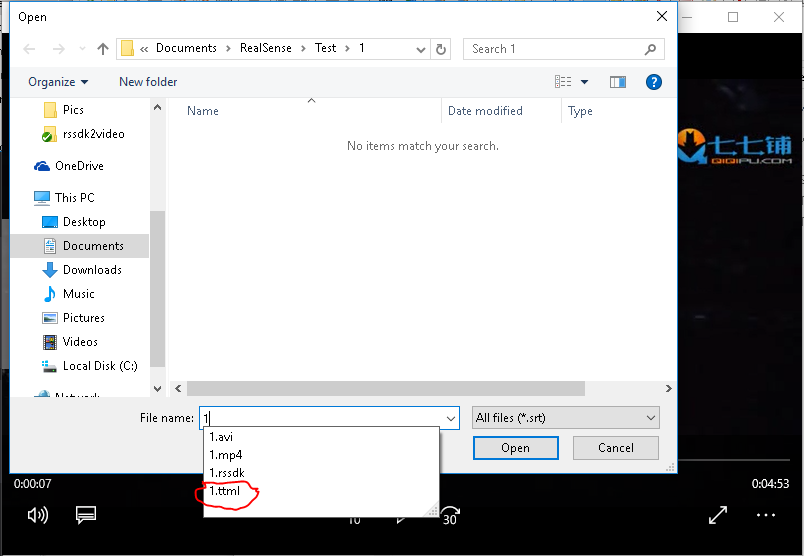
1. Go to the projects bin directory
2. Run EmoTracker.exe and set calibration file (can be obtained by calibrate process used FF\_EyeTracking or pressing **Calibrate** button on the form), output for stream from camera and emotions subtitles file

t.ex: 1.rssdk and 1.ttml

1. On can choose if application should collect gaze data, put plus as a marker of gaze point into the written subtitles, and should application use RS SDK Person Tracker module to making decision about emotion.
2. Click **Start** button
3. To finish recording press Stop button
4. As a result, it is a two files 1.rssdk with camera output record and 1.ttml with emotions track and gaze directions (if one not specify **RSSDK output**  then this file should not be written)

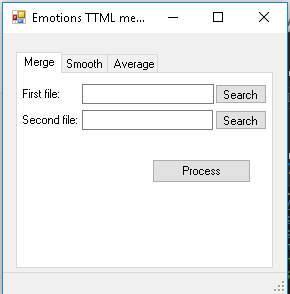
How to use results

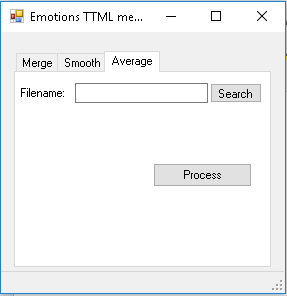
* ttml file contains timed to the presentation of text media in synchrony with other media, such as audio and video.(see https: //www.w3.org/TR/ttml2/) allowing to be represented in the form of subtitles to video content. One of the supporting such format players is the Windows 10 Movies & TV

1. Play video using the Movies & TV, for example, a video file, which was launched during the emotions tracking.
2. Choose subtitle
3.  When you select a file, files with the ttml extension will not be visible, you shoold explicitly specify the name of the file with subtitles
4. You can also use video recorded from the camera during emotions traking. To convert the recorded stream from rssdk format to the common used video format one can use rssdk2video utility. For example, having 1.rssdk file and assembled rssdk2video project, you can can create the video stream of 30 fps with a resolution of 480x270, by running the command line:

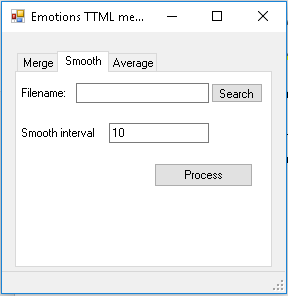
$(path to rssdk2video)\rssdk2video.exe 1.rssdk 1.avi 30 480 270

**Case 2 Merge emotions tracks**

1. Go to the projects bin directory
2. Run EmoMerge.exe to merging, averaging and smoothing emotions tracks
3. To merge two emotions tracks
   1. Press **Merge** tab
   2. Choose first and second files with emotions tracks by pressing **Search** button to merge them
   3. Press **Process** button and choose output file name
4. To averaging emotions data
   1. Press **Average** tab
   2. Choose file with emotions track by pressing **Search** button
   3. Press **Process** button and choose output file name



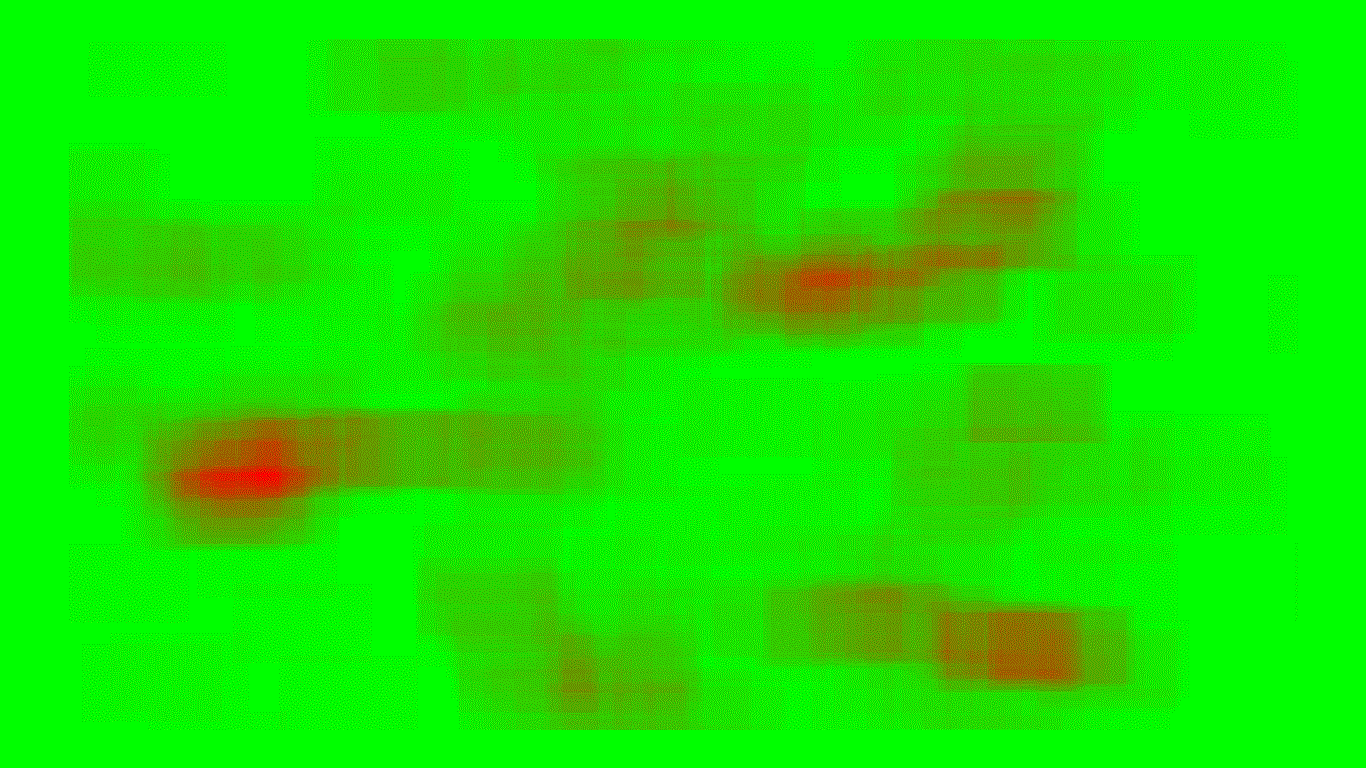
1. Emotions records with high frame rate and usually it more convenient to smooth it before using. To do this:
   1. Press **Smooth** tab
   2. Choose file with emotions track by pressing **Search** button
   3. Choose time interval to smooth
   4. Press **Process** button and choose output file name



**Case 3 Display and prepare gaze heat map**

1. Go to the projects bin directory
2. Run GazeHeatMap.exe and choose emotions track to build gaze heat map
3. Choose file name to store produced map
4. One can cover obtained heat map over application has been watched during emotions tracking



1. And see heat map picture
2. You can also use video watched during emotions traking. To put the recorded gaze area on to the video one can use GazePainter utility. For example, having 1.mp4 file and assembled GazePainter project, you can can create the video stream of 30 fps with a resolution of 480x270, by running the command line:

$(path to GazePainter)\GazePainter.exe 1.ttml 1.mp4 1.avi 30 480 270

**Contributing**

1. Fork it!
2. Create your feature branch: git checkout -b my-new-feature
3. Commit your changes: git commit -am 'Add some feature'
4. Push to the branch: git push origin my-new-feature
5. Submit a pull request :D

**History**

This is first release

**Credits**

Thanks to Intel for the funny days I have spent with its code.

**License**

I think it should be Copyleft.

But seems Intel want Apache License 2.0