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                # emotions writer prototype (used only for research
│                             # purposes)
├── library                   # EmoTracker libraries
│   ├── CSharpLibrary         # C# wrapper for native library
│   │   ├──Build.docx         # Build instructions
│   │   └──CSharpLibrary.sln  # VS 2015 solution
│   ├── emotracker            # Native C++ Windows library
│   │   ├──docs               # API documentation
│   │   ├──Build.docx         # Build instructions
│   │   ├──emotracker.sln     # VS 2015 solution
│   │   └── ...               # etc.
│   └── ...                   # etc.
├── rssdk2video               # Utility for converting rssdk format to video formats
│   │                         # (it should be in utils dir)
│   ├── Build.docx            # Build instructions
│   ├── rssdk2video.sln       # VS 2015 solution
│   └── ...                   # etc.
├── samples                   # Examples for using
│   ├── EmoMerge              # C# application to merge emotions TTML files
│   │   └──EmoMerge.sln       # VS 2015 solution
│   ├── EmoTracker            # C# application used EmoTracker library for emotions
│   │   │                     # recording
│   │   ├──Build.docx         # Build instructions
│   │   ├──EmoTracker.sln     # VS 2015 solution
│   │   └── ...               # etc.
│   └── ...                   # etc.
├── utils                     # Utilities to use emotions tracks
│   ├── GazeHeatMap           # Map recorded gaze onto the video
│   ├── GazePainter           # Map recorded gaze onto the video
│   └── ...                   # etc.
├── Doxyfile                  # Doxygen configuration file to build documentation
├── survey.doc                # About this project
├── survey.pdf                # About this project
```

```
├── test.ttml                    #  TTML emotions record sample
└── ...
```

Installation

Prerequisites

`emotracker`:

To build it you need:

- **Microsoft Visual Studio 2015**
- **Microsoft Visual C++ 2015**
- **Intel® RealSense™ SDK 2016 K2**

library/emotracker:

To build it you need:

- **Microsoft Visual Studio 2015**
- **Microsoft Visual C++ 2015**
- **Intel® RealSense™ SDK 2016 K2**

library/CSharpLibrary:

To build it you need:

- **Microsoft Visual Studio 2015**
- **Microsoft Visual C# 2015**
- **Intel® RealSense™ SDK 2016 K2**
- **emotracker library**

rssdk2video:

To build it you need:

- **Microsoft Visual Studio 2015**
- **Microsoft Visual C++ 2015**
- **Intel® RealSense™ SDK 2016 K2**
- **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 R2

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
    a. `Project -> Properties`
    b. `C/C++ -> General -> Additional include path` it should contain RSSDK include path: $(RSSDK_DIR)/include
3. Check path to RSSDK libraries directory
    a. `Project -> Properties`

    b. `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
    a. If not, then right click on the Reference -> Add reference… to open Reference manager
    b. 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
    a. 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
    a. `Project -> Properties`

   b. `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 $(OPENCV_DIR) points to correct OpenCV path
   a. To display **Property Manager**, on the menu bar, choose **View**, **Other Windows**, **Property Manager**.
   b. Expand rssdk2video -> Debug | Win32, right click on PropertySheet to open **Property Page** dialog
   c. Select **User Macros** tab, and verify correctness of `OPENCV_DIR` macros defines the path to OpenCV library location
4. Check path to RSSDK libraries directory
   a. `Project -> Properties`
   b. `Linker -> General -> Additional library path` it should contain RSSDK and OpenCV library path, t.ex.:
   `$(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`:

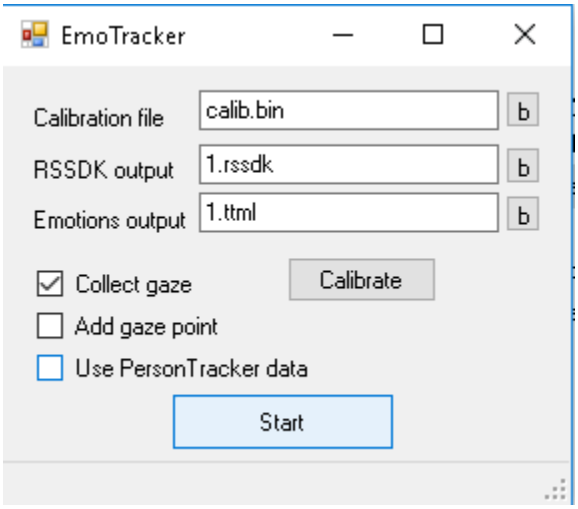
1. Open GazeHeatMap.sln with Microsoft Visual Studio 2015
2. Build and save the library `GazeHeatMap.exe`, under the local `bin` directory

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 form), output for stream from camera and emotions subtitles file

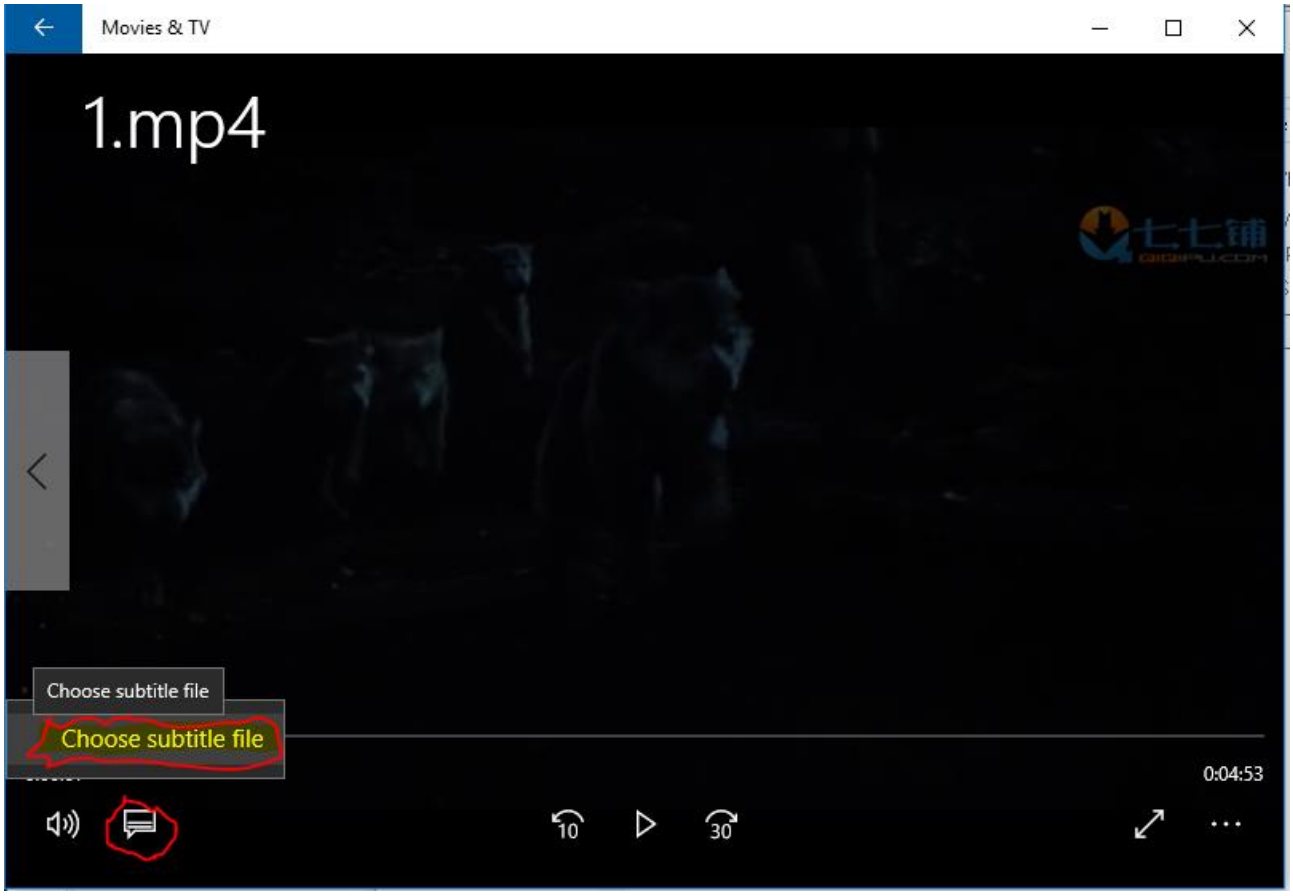


t.ex: `1.rssdk` and `1.ttml`

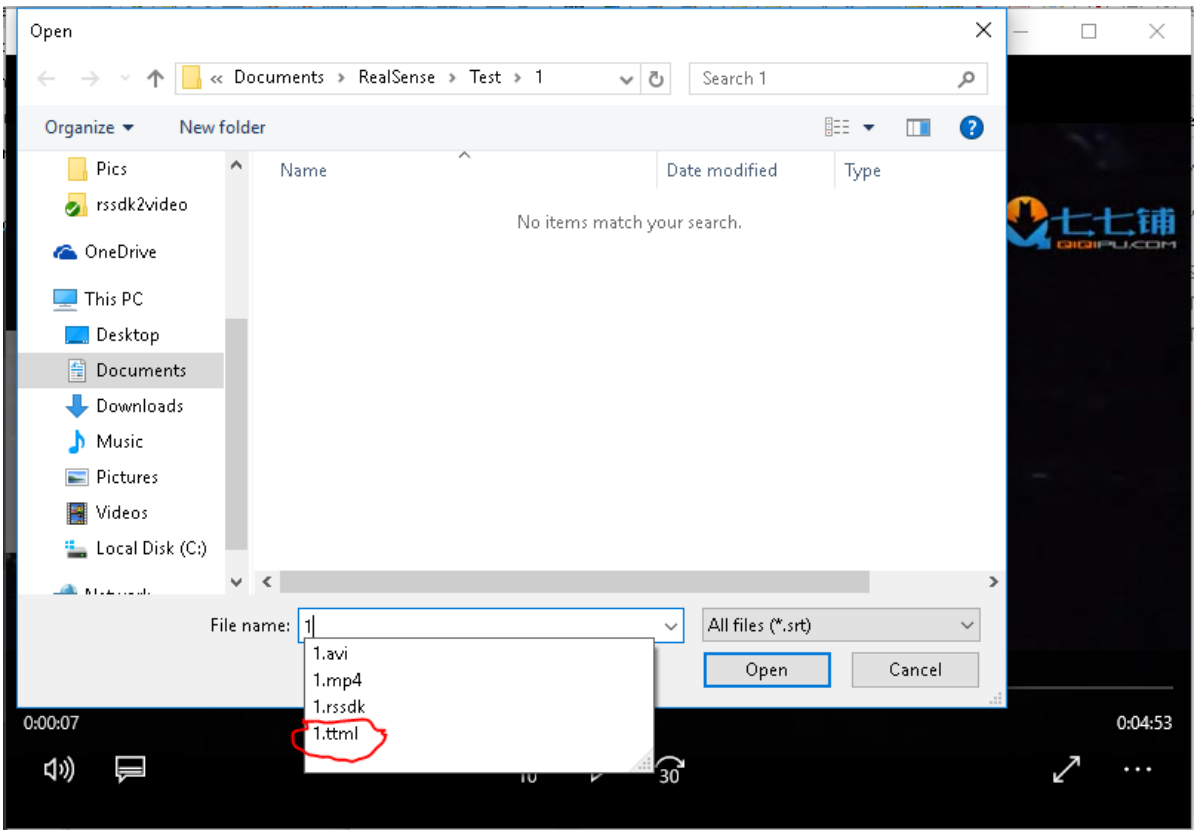
3. On can choose if application should collect gaze data, put plus as a marker of gaze point into the written subtitles, and should application use RS SDK Person Tracker module to making decision about emotion.
4. Click **`Start`** button
5. To finish recording press `Stop` button
6. As a result, it 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
```

5.

Case 2

Case 3

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