Realtime Breathing
Respiratory rate and chest movement analysis
Nili Furman & Maayan Ehrenberg
Goals And Motivation

• Investigate how artificial intelligence can be applied to visual information in order to detect abnormalities in chest motion.

• Introduce a simple and non-invasive procedure to measure respiratory rate of human patients, using a video camera.

• Contribute to Intel Realsense documentation and collection of examples.
Tools And Environments

- Intel Realsense
- Github source code & examples
- SDK 2.0

- Intel Realsense Depth Camera D435

- Visual Studio 2017
- C++
- CMake
- Dear ImGui
- OpenCV
- cv-plot
Methods

- Data Collection
- Image Processing
- Math
Data Collection

Assumptions:

• 4 or 5 stickers of ~2cm diameter
  • Supported colors: yellow, blue and green
  • Diameter may vary but must be consistent for all stickers
• Positioned accordingly
• Good visibility on chest
  • Required for color distinction
Image Processing
Image Processing (Detail)

- **Frameset** ~25 fps
  - Depth
  - Color
- **Wait** mechanism
Image Processing (Detail)

- Align depth to color
- Sizes difference
- Allow uniform pixel coordinates

Frame Polling → Alignment
Image Processing (Detail)

- RGB to BGR
  - OpenCV format
- BGR to HSV
  - Wider and more accurate selected color range
- Binary mask
- BGR to GRAY
- Image threshold
  - Binary B&W
  - Input for CC
Image Processing (Detail)

- `connectedComponentsWithStats()`
  - Returns centroids
  - Saved in FrameData

- Noise...
Image Processing (Detail)

- Set of last frames processed
  - Cleanup
    - Missing frames
    - Poor detection
  - Distances calculation

Steps:
- Frame Polling
- Alignment
- Color Detection
- Connected Components
- Frame Manager
Intel SDK – Frame Polling & Alignment

- **rs2::pipeline**
  - start()
  - stop()
  - wait_for_frames()

- **rs2::config**
  - enable_stream()
  - disable_stream()
  - RS2_STREAM_DEPTH or RS2_STREAM_COLOR

- **rs2::align**
  - align_to_depth(RS2_STREAM_DEPTH)
  - align_to_color(RS2_STREAM_COLOR)

- **rs2::frameset**
  - get_depth_frame \(\rightarrow\) rs2::depth_frame
  - get_color_frame \(\rightarrow\) rs2::video_frame
  - colorizer
  - Each frame has profile & ID
Intel SDK – Frame Manager

• `depth_frame.get_distance(x, y)`
• 3D distances calculation
  • `rs2_deproject_pixel_to_point(...)`
    (rs-measure example)
• `get_timestamp()`
Intel SDK – Rendering

- map<int, rs2::frame> render_frames
  - Stores frames to be rendered
  - render_frames[frame_id] = \texttt{colorizer.process}(frame)
- Class \texttt{window} (from \texttt{example.hpp})
  - \texttt{show}(rs2::frame)
Data Extraction

- Frames IDs
- Frames timestamps (Color & depth)
- Stickers coordinates in pixels (2D & 3D) and cm (3D only)
- 2D & 3D Distances $\Rightarrow$ 2D & 3D means
Math

• Fourier Transform
• Data from 256 frames “intervals” overlapping
• Frequency of highest peak (after noise reduction)
Math – 2D Sample
Math – 3D Sample
Our Resolutions
Intel SDK, Viewer and Examples

• Intel RealSense Viewer
• Provides a wide set of camera configurations and properties
• Nice GUI
• Sophisticated Implementation

NO DOCUMENTATION

CREATING BASIC GUI EXAMPLES BY OURSELVES
Hough vs. Connected Components

- Hough Algorithm to find circles
  - Not robust enough
  - Requires multiple (expensive) iterations to work correctly

- Solution: Connected Components Algorithm (with stats)
  - Robust and efficient
  - Finds centroids
  - But...
    - Noise sensitive
    - Relies on fine color detection
Color Detection – HSV

• RGB color ranges are too strict
  • No feasible ranges for a color considering its shades and highlights

• Hue, Saturation, Value
  • Allows wide selection of yellows (and blues, greens...)
  • Wide shades and saturation range
  • Pretty precise
Color Detection – HSV

- For yellows: \((20, 50, 50) \rightarrow (40, 255, 255)\) (OpenCV)
- Blues: \((90, 20, 10) \rightarrow (135, 255, 255)\)
- Greens: \((35, 30, 30) \rightarrow (85, 255, 255)\)
Fourier Transform – Noises

• Filters (hpf, lpf...)
  • Avoid aliasing
  • Results remained unaffected – aliasing was not the cause

• Alternative frequency determination method: average time elapsed between peaks
  • Works well in 2D
  • Extensively affected by parameters fine tuning
    • More data is needed to validate current tuning

• **Samples normalization**
  • Each iteration, according to current max and min values
  • Normalized and shifted to [-1, 1]
  • Results improved significantly
Results

• 2D measurements are quite precise, but prone to small error.
• 2D implementation is based on color detection and can be altered to be used in other platforms.
• 3D measurements are noisier and thus less reliable. Depth is not always achieved via the 3D camera (might be improved by usage of a wider depth range and more precise depth camera).
• 3D measurements supply subtle enough depth information for abnormal chest movement, and further processing can be done to detect abnormal motion. (anti-phase breathing, breath volume, etc.)
• Fourier transform might be insufficient for BPM calculation in 3D due to non-negligible noises.
Expected vs Results

• Camera and manual measurements will be identical (up to a small error extent). Although camera measurements are valid only under certain conditions.

• RGB measurements will be sufficient at certain circumstances; Implementation can be relevant for other platforms.

• 3D measurements might supply additional information and assist in case of non-favorable conditions (frame disturbances).

• 3D measurements are more prone to noises and disrupts and supply less legible information.

• Lay a footprint on Realsense Github! 😊 – 😞
Future Work

- Breath Volume
- Phase Shift
- Remove stickers dependency
Questions?