Gesture recognition with Kinect

Joakim Larsson

Outline

- Task description
- Kinect description
- AdaBoost
- Building a database
- Evaluation

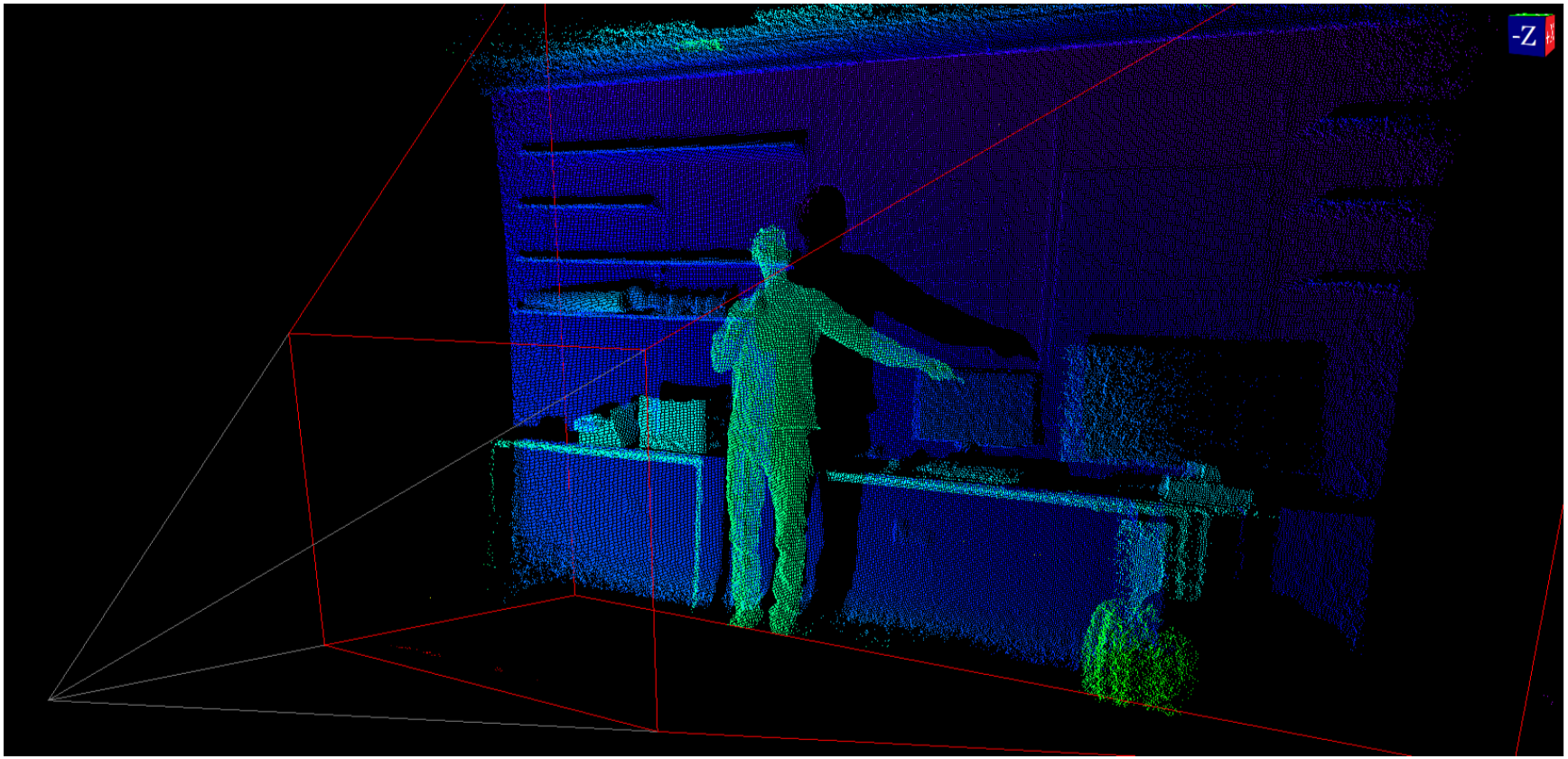
Task Description

- The task was to implement gesture detection for some gestures using a Kinect sensor.
- The gesture chosen was if the user was pointing in a direction.

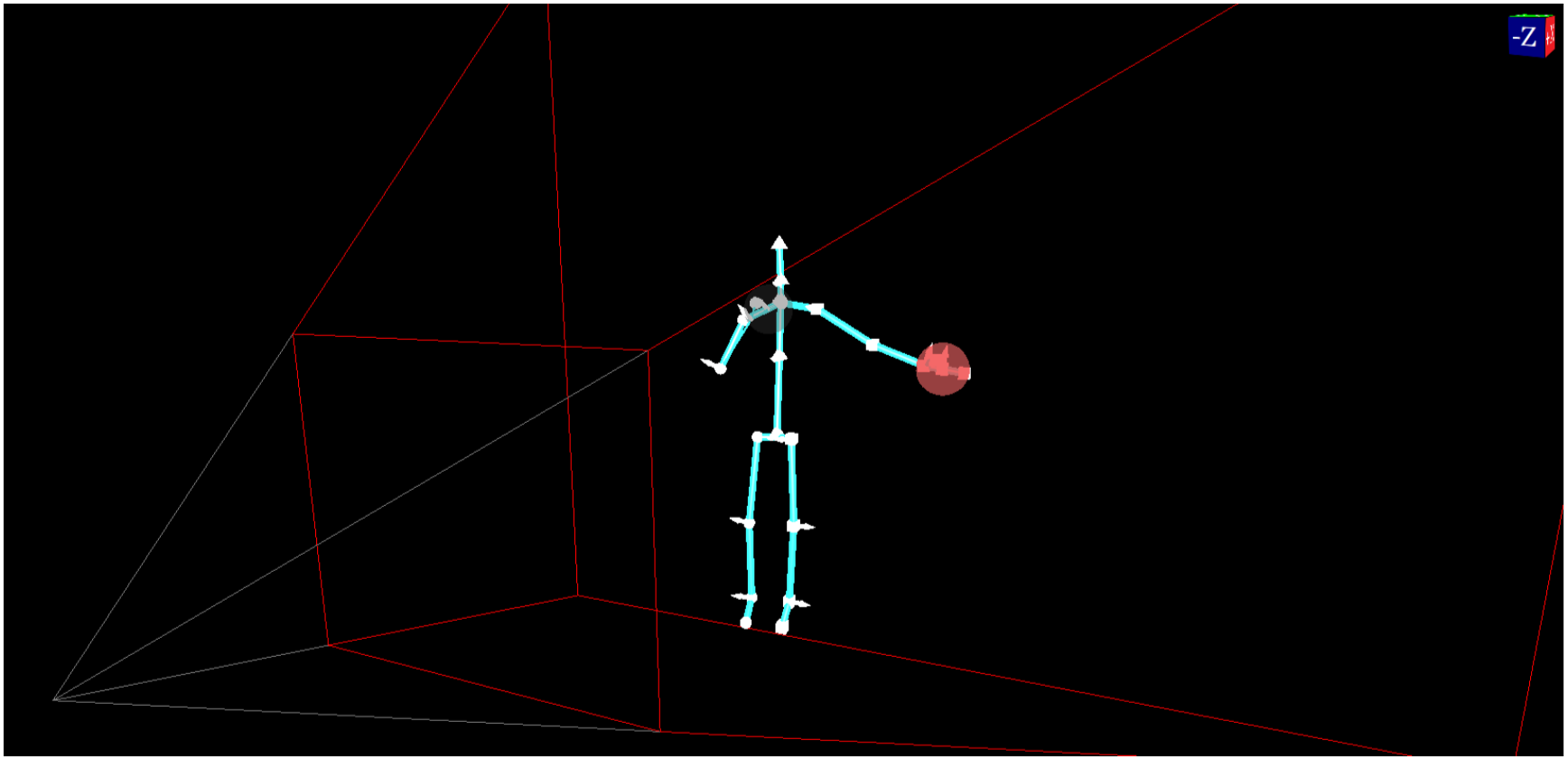
Kinect

- Kinect is a motion sensing input device.
- It records the distance between the sensor and any objects in its way as a point cloud.
- From these points there exists software to extract a human body from the point cloud live.

Kinect



Kinect



First solution

- The first solution used was to extract the joints and determine if the hand-, wrist-, elbow-joints' position for both arms was close to forming a line.
- In such a scenario the user would be considered to be pointing.
- This solution required the user to hold their arms VERY straight however.

- The method eventually chosen uses the meta algorithm AdaBoost.

AdaBoost

- AdaBoost stands for Adaptive boosting.
- It won the Gödel Prize in 2003.
- It’s an algorithm with which a machine can be taught what classifiers are significant when determining whether a discrete event has occurred or not.

AdaBoost example

- Imagine we wish to classify a four number string on an unknown criterion.
- The only thing we know of the criterion is the information we can gather from a set of training examples.

AdaBoost Example

STRING	CLASS
0001	FALSE
0012	FALSE
0415	FALSE
0881	FALSE
0888	TRUE
1234	FALSE
1235	FALSE
1299	TRUE
1515	FALSE
1559	FALSE
7654	TRUE
7771	TRUE
7777	TRUE
7779	TRUE
7780	TRUE
8337	TRUE
8502	FALSE
9001	FALSE
9039	TRUE
9999	TRUE

- In order to classify the training set some simple classifiers are needed.
- In this example case the classifiers are of two natures:
  - Treating the strings as integers and asking whether they are above or below a certain threshold value.
  - Counting how many of the integers in the string that are of a certain type.

AdaBoost Example

STRING	CLASS	>7650
0001	FALSE	FALSE
0012	FALSE	FALSE
0415	FALSE	FALSE
0881	FALSE	FALSE
0888	TRUE	FALSE
1234	FALSE	FALSE
1235	FALSE	FALSE
1299	TRUE	FALSE
1515	FALSE	FALSE
1559	FALSE	FALSE
7654	TRUE	TRUE
7771	TRUE	TRUE
7777	TRUE	TRUE
7779	TRUE	TRUE
7780	TRUE	TRUE
8337	TRUE	TRUE
8502	FALSE	TRUE
9001	FALSE	TRUE
9039	TRUE	TRUE
9999	TRUE	TRUE

- Examination shows that in this example 7650 is the best threshold value.

AdaBoost Example

STRING	CLASS	>7650	CORRECT
0001	FALSE	FALSE	TRUE
0012	FALSE	FALSE	TRUE
0415	FALSE	FALSE	TRUE
0881	FALSE	FALSE	TRUE
0888	TRUE	FALSE	FALSE
1234	FALSE	FALSE	TRUE
1235	FALSE	FALSE	TRUE
1299	TRUE	FALSE	FALSE
1515	FALSE	FALSE	TRUE
1559	FALSE	FALSE	TRUE
7654	TRUE	TRUE	TRUE
7771	TRUE	TRUE	TRUE
7777	TRUE	TRUE	TRUE
7779	TRUE	TRUE	TRUE
7780	TRUE	TRUE	TRUE
8337	TRUE	TRUE	TRUE
8502	FALSE	TRUE	FALSE
9001	FALSE	TRUE	FALSE
9039	TRUE	TRUE	TRUE
9999	TRUE	TRUE	TRUE
			0,8
			0,2

- Examination shows that in this example 7650 is the best threshold value.
- This labels 80 % of the examples correctly.

AdaBoost Example

STRING	CLASS	>7650	CORRECT	#0>0	CORRECT
0001	FALSE	FALSE	TRUE	TRUE	FALSE
0012	FALSE	FALSE	TRUE	TRUE	FALSE
0415	FALSE	FALSE	TRUE	TRUE	FALSE
0881	FALSE	FALSE	TRUE	TRUE	FALSE
0888	TRUE	FALSE	FALSE	TRUE	TRUE
1234	FALSE	FALSE	TRUE	FALSE	TRUE
1235	FALSE	FALSE	TRUE	FALSE	TRUE
1299	TRUE	FALSE	FALSE	FALSE	FALSE
1515	FALSE	FALSE	TRUE	FALSE	TRUE
1559	FALSE	FALSE	TRUE	FALSE	TRUE
7654	TRUE	TRUE	TRUE	FALSE	FALSE
7771	TRUE	TRUE	TRUE	FALSE	FALSE
7777	TRUE	TRUE	TRUE	FALSE	FALSE
7779	TRUE	TRUE	TRUE	FALSE	FALSE
7780	TRUE	TRUE	TRUE	TRUE	TRUE
8337	TRUE	TRUE	TRUE	FALSE	FALSE
8502	FALSE	TRUE	FALSE	TRUE	FALSE
9001	FALSE	TRUE	FALSE	TRUE	FALSE
9039	TRUE	TRUE	TRUE	TRUE	TRUE
9999	TRUE	TRUE	TRUE	FALSE	FALSE
			0,8		0,35
			0,2		0,65

- The next criterion would be to see how many zeroes are in the string.
- After that ones.
- And so on.

AdaBoost Example

STRING	CLASS	Weight	>7650	CORRECT	#0>0	CORRECT	#0>1	CORRECT	#1>0	CORRECT
0001	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
0012	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
0415	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
0881	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
0888	TRUE	0,05	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
1234	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1235	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1299	TRUE	0,05	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE
1515	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1559	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
7654	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7771	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE
7777	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7779	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7780	TRUE	0,05	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
8337	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8502	FALSE	0,05	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
9001	FALSE	0,05	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
9039	TRUE	0,05	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
9999	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		1		0,8		0,35		0,35		0,15
				0,2		0,65		0,65		0,85

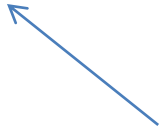
AdaBoost Example

- The classifier that give the smallest error, i.e. the one that for the training set returns the correct classification most often is chosen as our primary classifier.

AdaBoost Example

STRING	CLASS	Weight	>7650	CORRECT	#0>0	CORRECT	#0>1	CORRECT	#1>0	CORRECT
0001	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
0012	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
0415	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
0881	FALSE	0,05	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
0888	TRUE	0,05	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
1234	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1235	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1299	TRUE	0,05	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE
1515	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
1559	FALSE	0,05	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
7654	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7771	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE
7777	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7779	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7780	TRUE	0,05	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
8337	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8502	FALSE	0,05	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
9001	FALSE	0,05	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
9039	TRUE	0,05	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
9999	TRUE	0,05	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
		1		0,8		0,35		0,35		0,15
				0,2		0,65		0,65		0,85

- In the example using #1= 0 as a classifier nets 3/20 incorrect labelings.



AdaBoost Example

- In order to determine how well the classifier worked we assign it an α-value dependent on its error rate.
- $\alpha_t = \frac{1}{2} \cdot \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$

AdaBoost Example

- We now want to find additional classifiers
- Since our current classifier classifies some examples quite well, but others not so much we want our next classifier to primarily classify those incorrectly labeled correctly.
- The relative weight of the different examples are thus multiplied by e^{α} if the classifier classified incorrectly and $e^{-\alpha}$ otherwise.

AdaBoost Example

STRING	CLASS	Weight
0001	FALSE	0,015
0012	FALSE	0,015
0415	FALSE	0,015
0881	FALSE	0,015
0888	TRUE	0,015
1234	FALSE	0,015
1235	FALSE	0,015
1299	TRUE	0,085
1515	FALSE	0,015
1559	FALSE	0,015
7654	TRUE	0,015
7771	TRUE	0,085
7777	TRUE	0,015
7779	TRUE	0,015
7780	TRUE	0,015
8337	TRUE	0,015
8502	FALSE	0,085
9001	FALSE	0,015
9039	TRUE	0,015
9999	TRUE	0,015

AdaBoost Example

- We then continue by picking out the criterion that minimizes the weighted errors.
- Afterwards, we calculate the corresponding α-value, and calculate the new weights for our training set.
- Afterwards, the best weighted criterion is calculated.
- This continues until we have a sufficiently good combined classifier.

AdaBoost Example

STRING	CLASS	Weight	>1240	CORRECT	#0>0	CORRECT	#0>1	CORRECT
0001	FALSE	0,015	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE
0012	FALSE	0,015	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE
0415	FALSE	0,015	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE
0881	FALSE	0,015	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE
0888	TRUE	0,015	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
1234	FALSE	0,015	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
1235	FALSE	0,015	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE
1299	TRUE	0,085	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
1515	FALSE	0,015	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
1559	FALSE	0,015	TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
7654	TRUE	0,015	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
7771	TRUE	0,085	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
7777	TRUE	0,015	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
7779	TRUE	0,015	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
7780	TRUE	0,015	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
8337	TRUE	0,015	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
8502	FALSE	0,085	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
9001	FALSE	0,015	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
9039	TRUE	0,015	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
9999	TRUE	0,015	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE
		1		0,715686		0,205882		0,343137
				0,284314		0,794118		0,656863

- In the example using #0> 0 as a classifier nets a better weighted value, despite the numerical thresholding correctly classifying more examples.

AdaBoost Example

- The final classifier for examples x outside the training set is then given by the sign of:
- $\sum_{k}^{k_{max}} \alpha_k \cdot h_k(x)$
- where $h_k(x)$ is the k:th classifier returning +1 for passing classification and -1 otherwise.

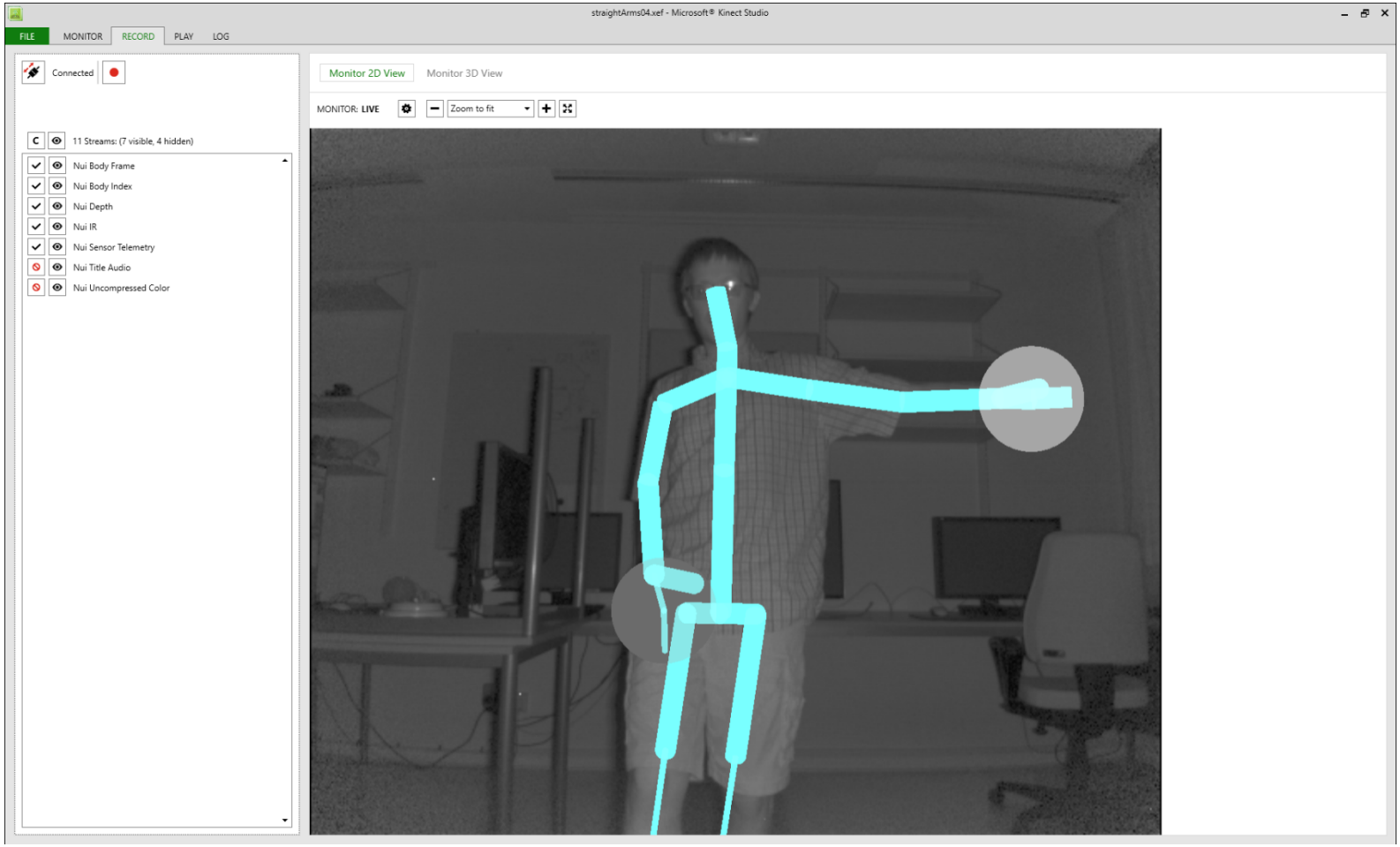
AdaBoost with Kinect

- This is the algorithm that is in use in the gesture detection software that comes with the Kinect SDK.
- While the example used strings of integers, the Kinect SDK uses the reconstructed skeleton from the point cloud as training data.
- From this it extracts features, such as angles, difference in position, speed, etc.

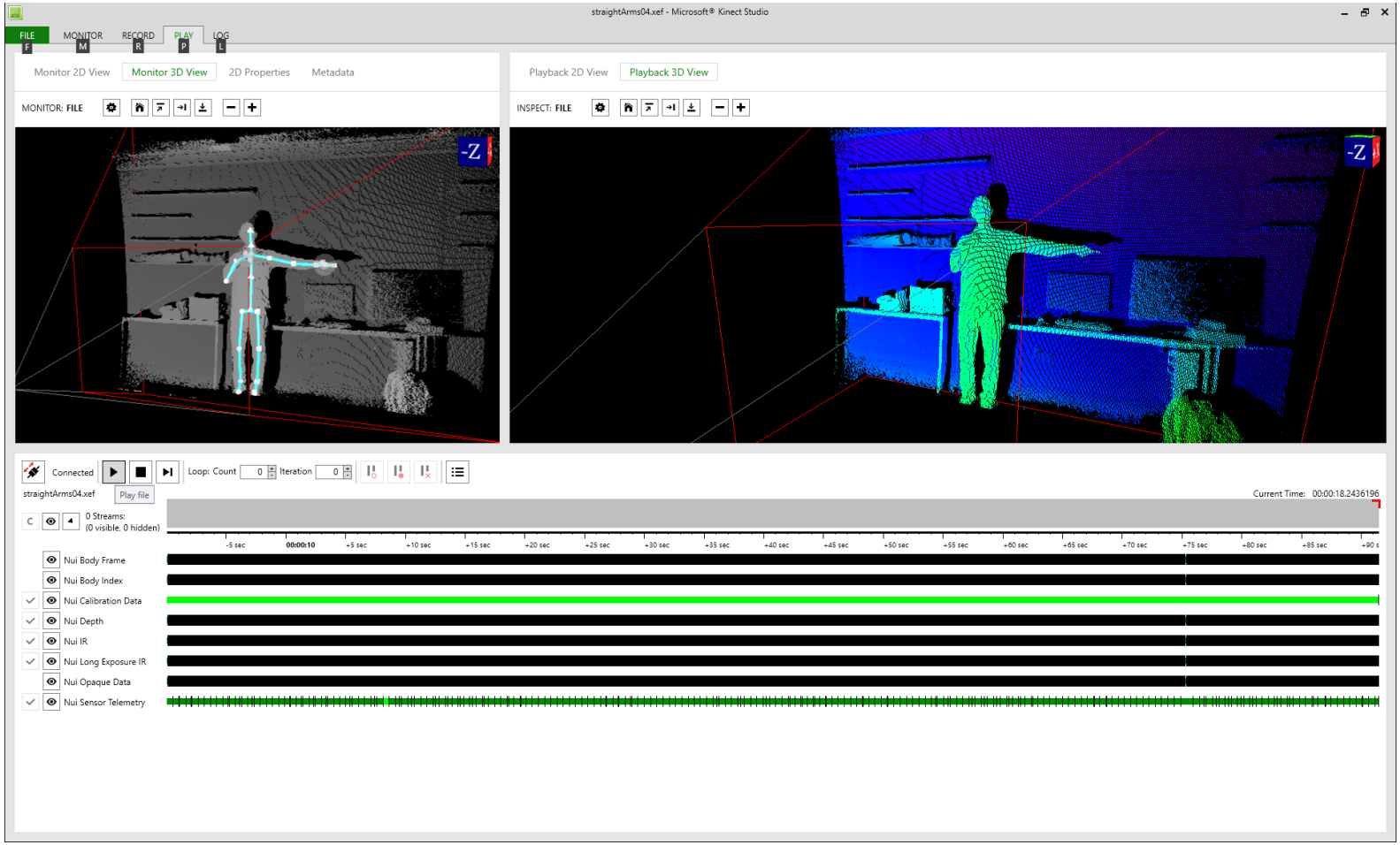
Using the Kinect SDK

- The Kinect SDK comes with a wealth of software to create a database.
- First is Kinect Studio, with which one can capture footage from the Kinect.
- Second is Visual Gesture Builder, with which one can train a classifier.
- Third is Visual Gesture Builder Viewer, with which one can verify how well the classifier works.

Kinect Studio



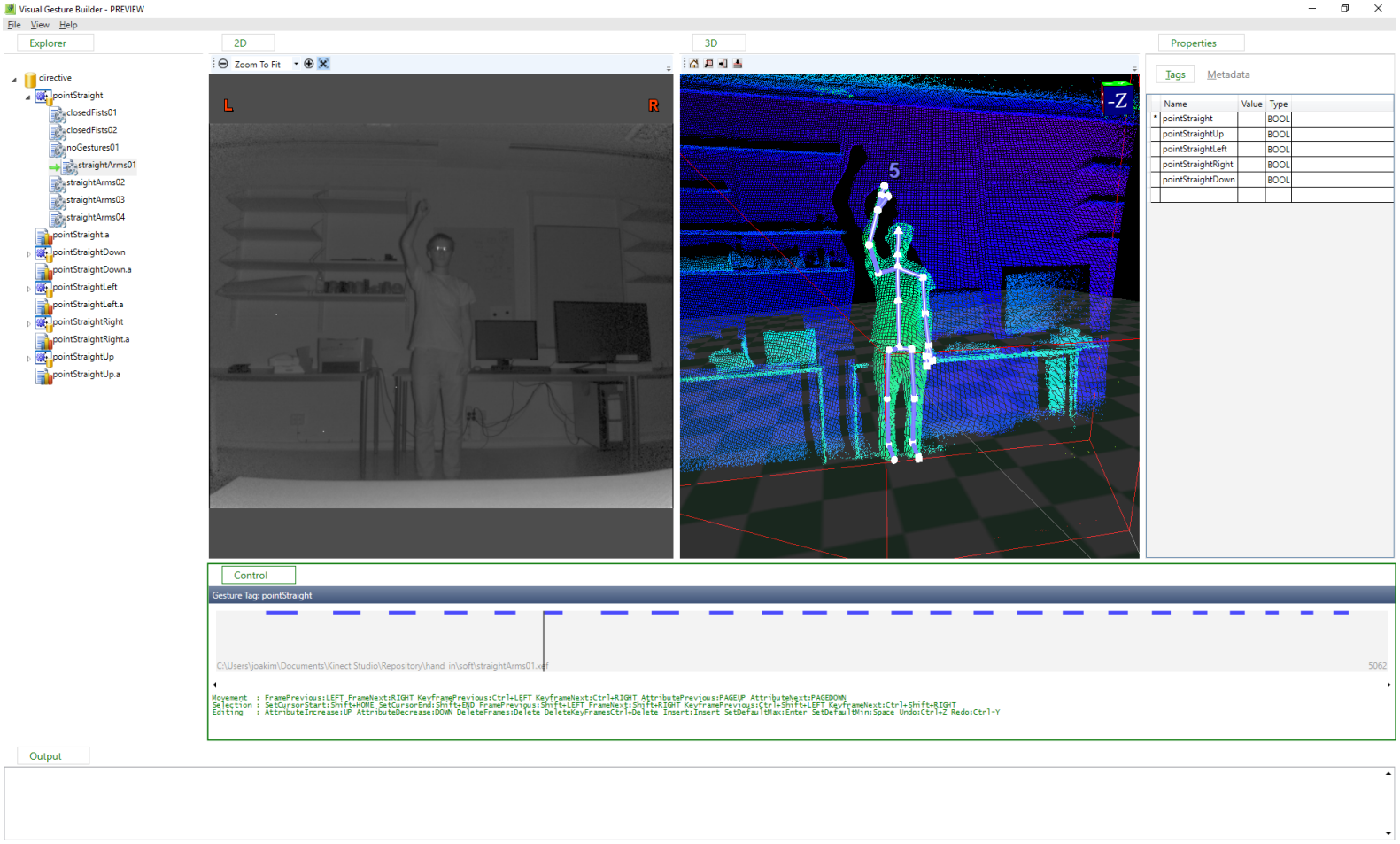
Kinect Studio



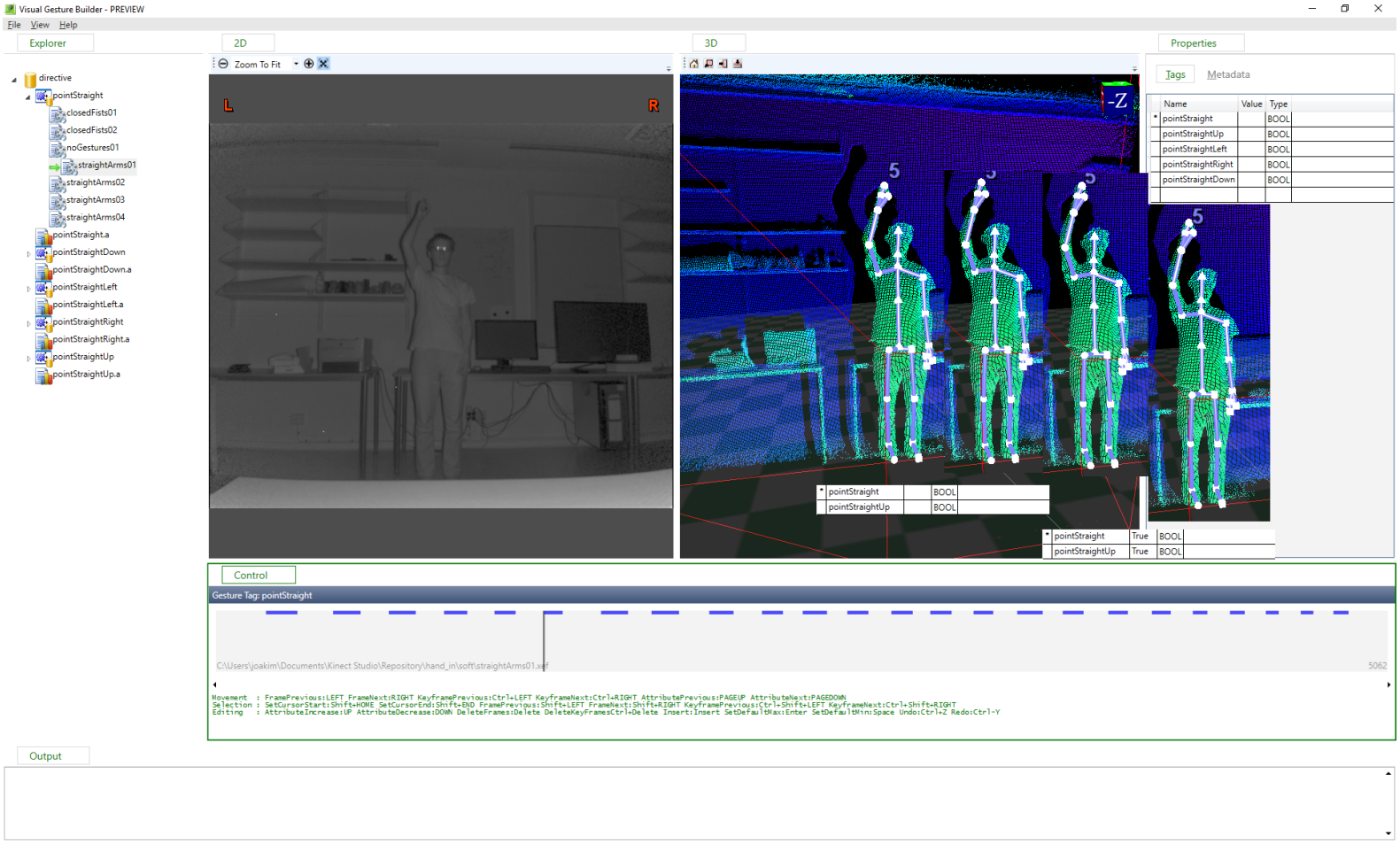
Visual Gesture Builder

- In the builder each captured frame of footage is considered one example in the training set.
- These frames are then tagged as fulfilling/not fulfilling the criterion.
- The Builder is then used to build a database, i.e. determine what classifiers best determine whether a gesture is active or not.

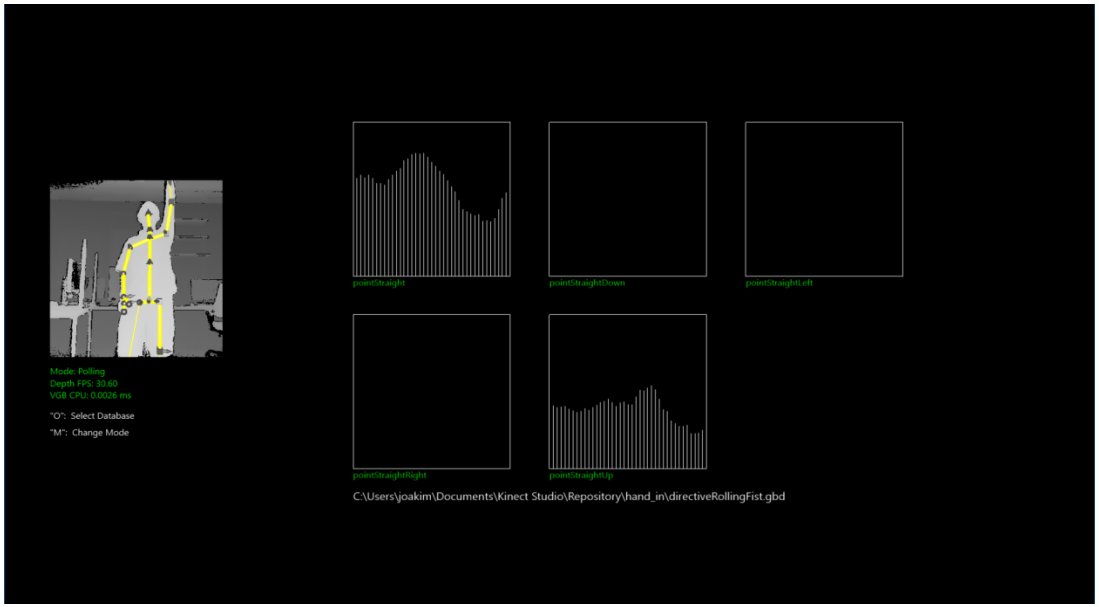
Visual Gesture Builder Viewer



Visual Gesture Builder



Visual Gesture Builder Viewer



- The Visual Gesture Builder Viewer is then used to examine how well the database classifies gestures.

Approach to the problem

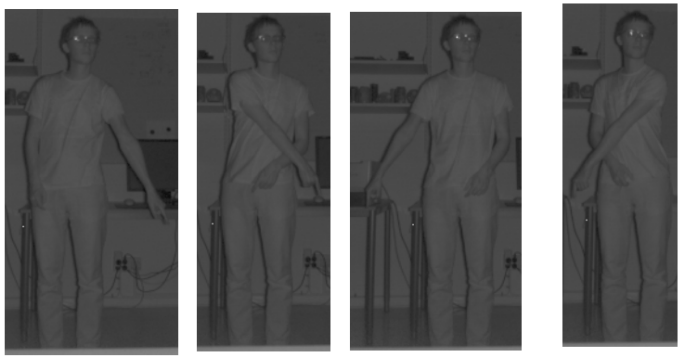
- The approach used was thus:
  - Capture footage with Kinect Studio where the gestures were performed.
  - Tag the frames of the footage during which the gestures were performed.
  - Let the Gesture Builder extract the appropriate features and build a database using AdaBoost.
  - Evaluate its accuracy with the Gesture Builder Viewer.
  - Repeat until the end of term.

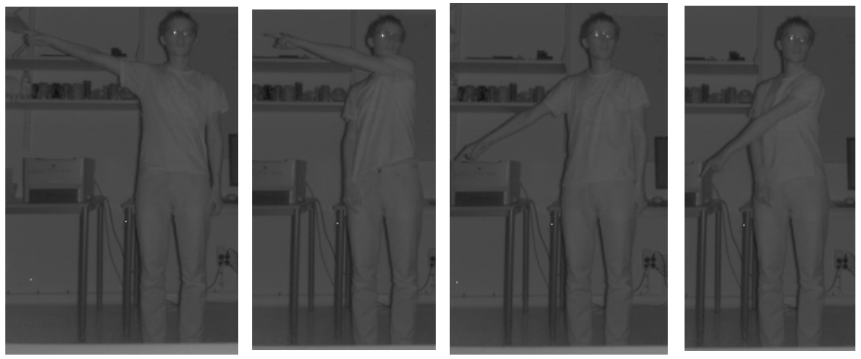
Approach

- The gesture to be captured was if the user is pointing in one of the four directions Up, Down, Left or Right. Additionally there was one generic gesture just for pointing.
- The first clip was of me pointing straight in these directions six times, three per hand.

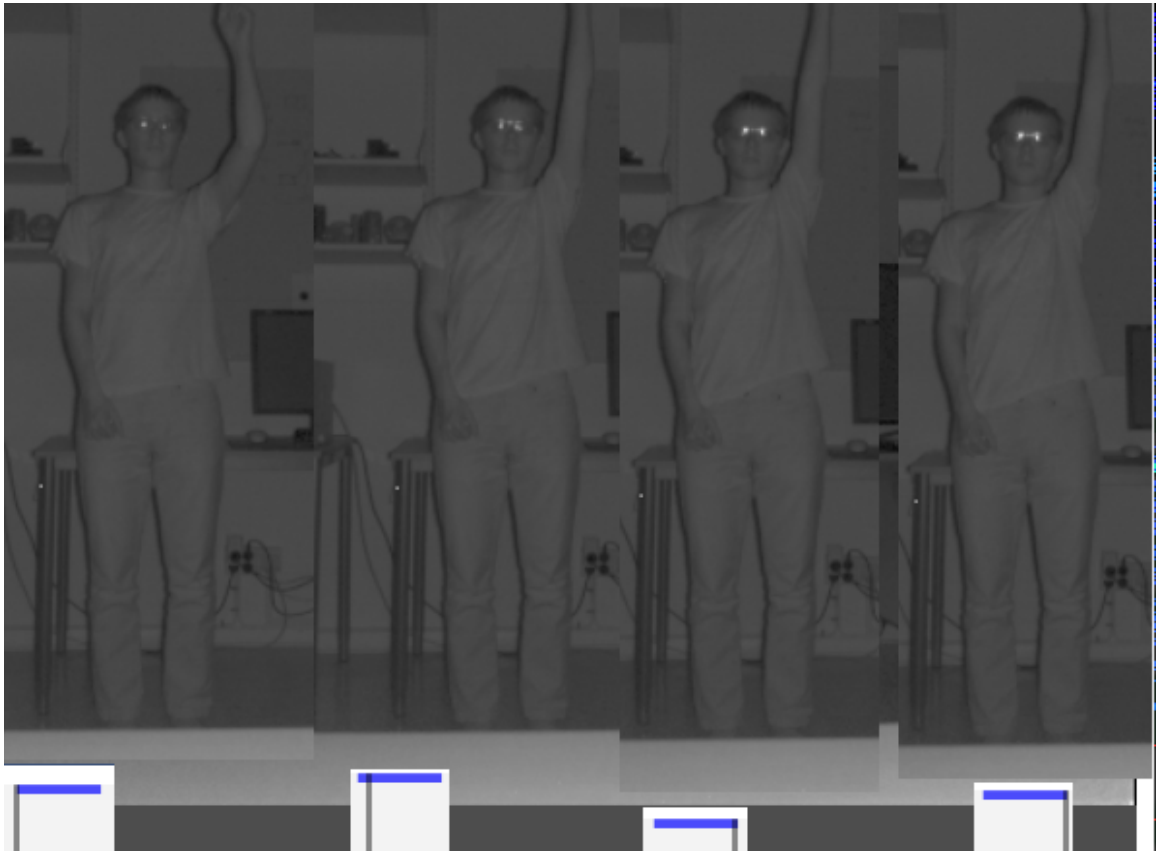
Features after first clip

- After the first clip, the #1 features discovered in each direction was:
  - Up: Angles(WristLeft, Head, WristRight)>=180
  - Down: Angles(ThumbRight, HandRight, WristRight)>=130
  - Left: MuscleTorqueZ(SpineShoulder)>=2
  - Right: Angles(Head, ShoulderRight, SpineBase)<126
- I continued adding clips in this fashion, so that the features used would be more correctly used.





- At this point I discovered that it was physically exhausting to stand in the positions the classifier demanded.
- I therefore went back through the already tagged clips to extend the time the arm would be considered pointing.



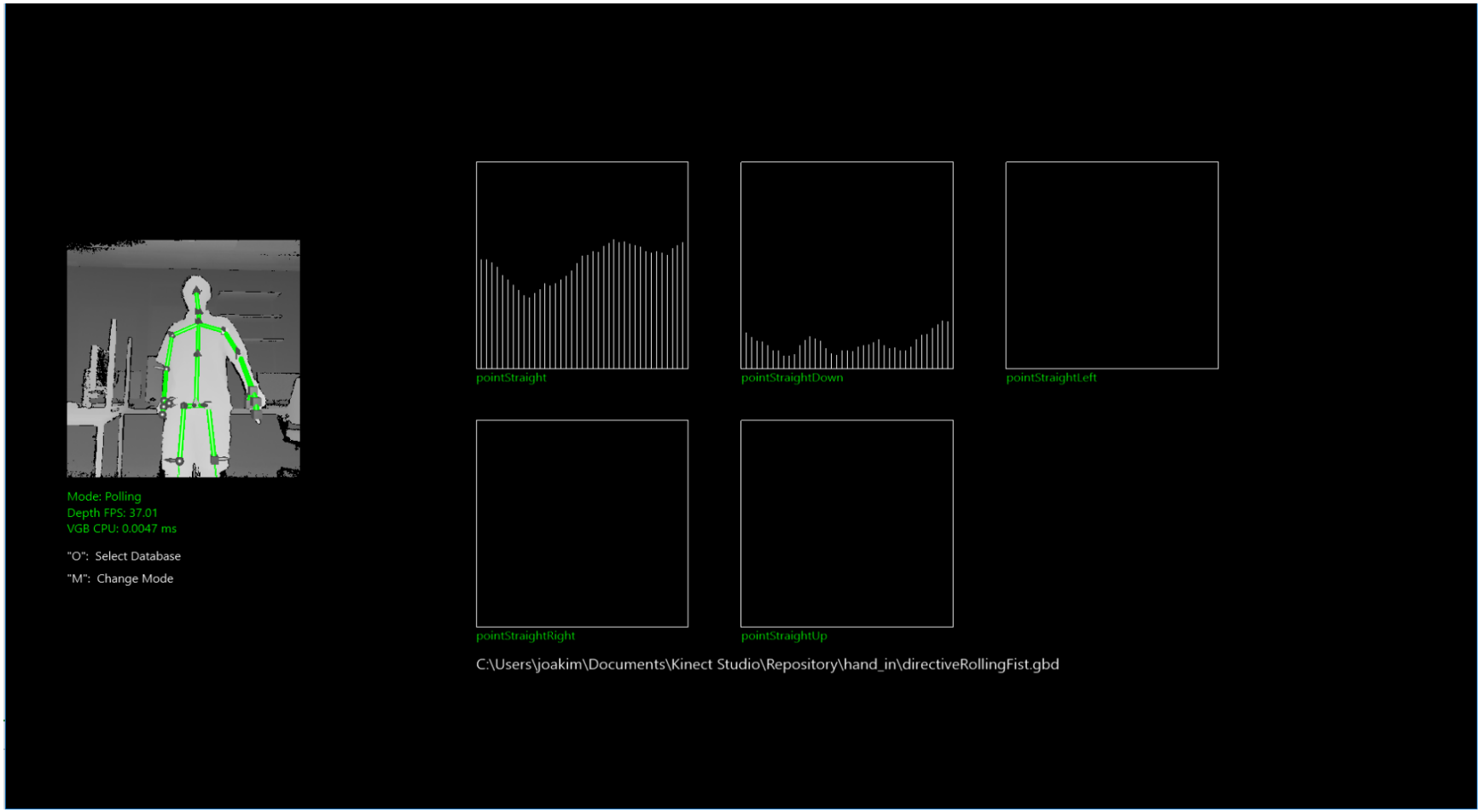
In the above up pointing movement, the middle ones used to be the start and finish of the frames tagged pointStraightUp.

- This continued until the term ended.
- In the final form there are two databases.
  - One which has been trained to regard a closed fist as not pointing.
  - One which has not.
- This is due to Kinect's inbuilt handState being quite uncertain when testing it online.

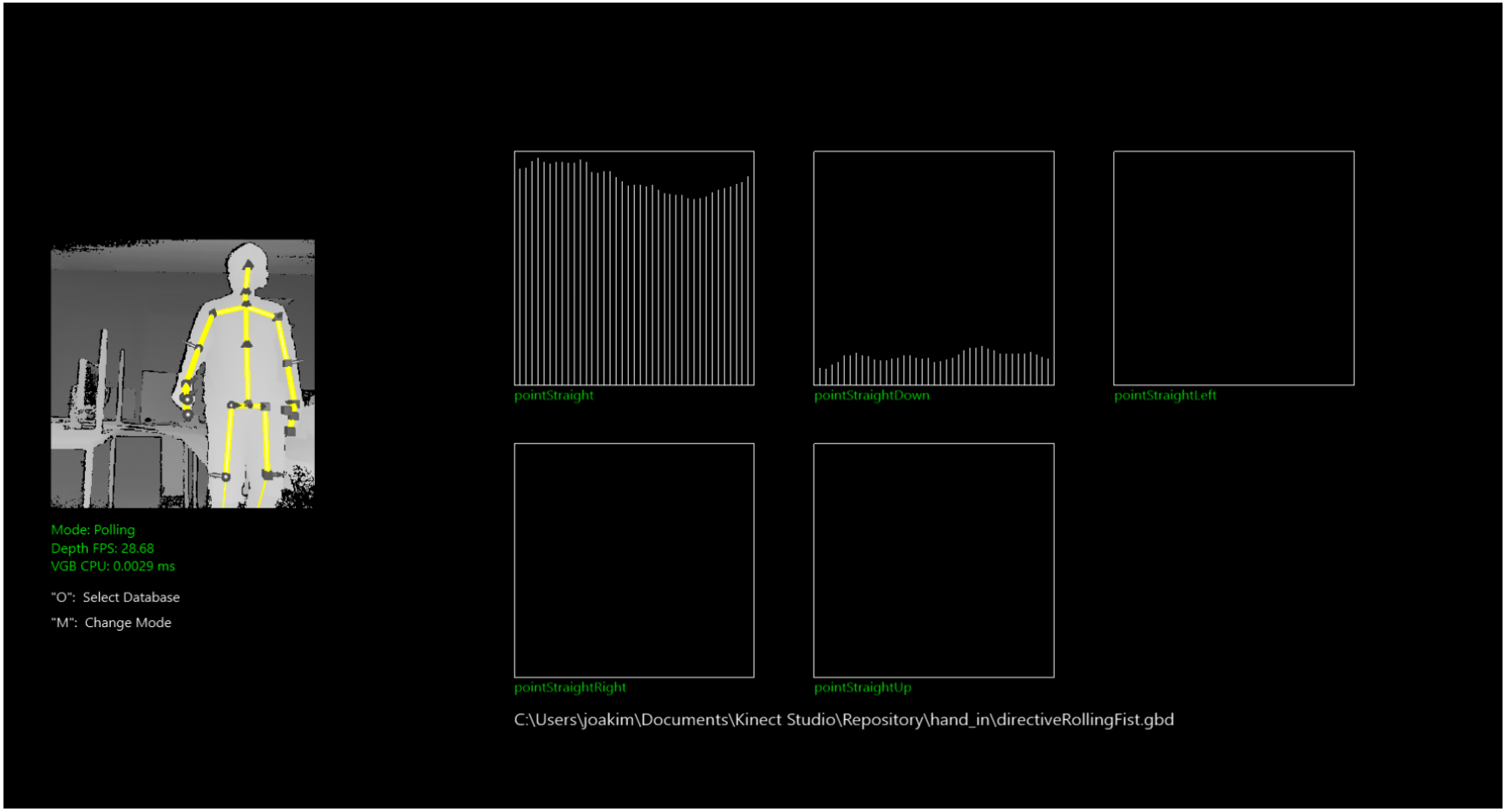
Evaluation

- The final database is, according to my personal evaluations, quite accurate in realizing that the user is pointing and in what direction.
- It is less certain in some directions.
- In the database that is supposed to ignore a closed fist the database often fail with that very task, so further training would be necessary.
- How successful the database is can vary.

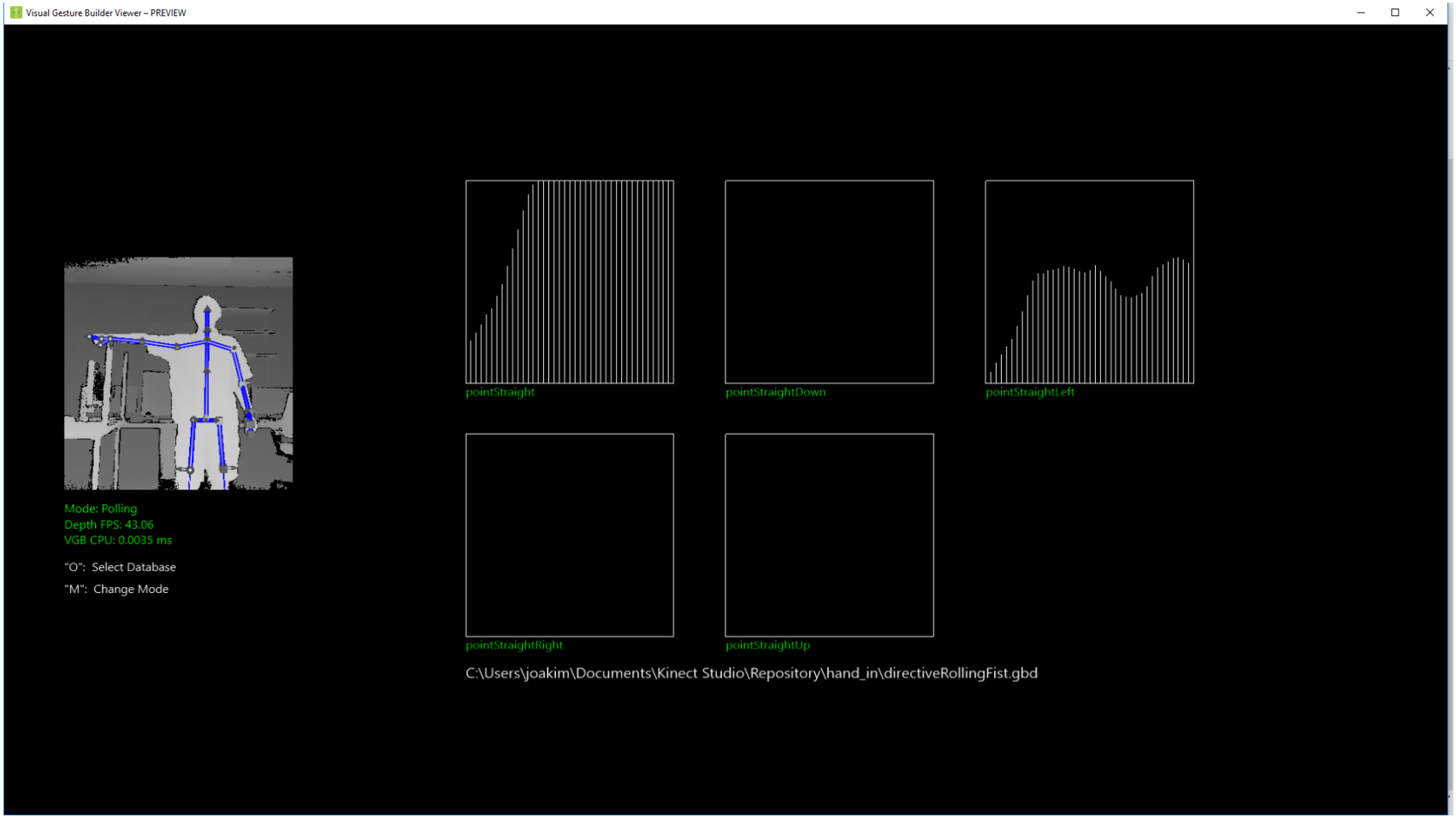
Point down


Mode: Polling
Depth FPS: 37.01
VGB CPU: 0.0047 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

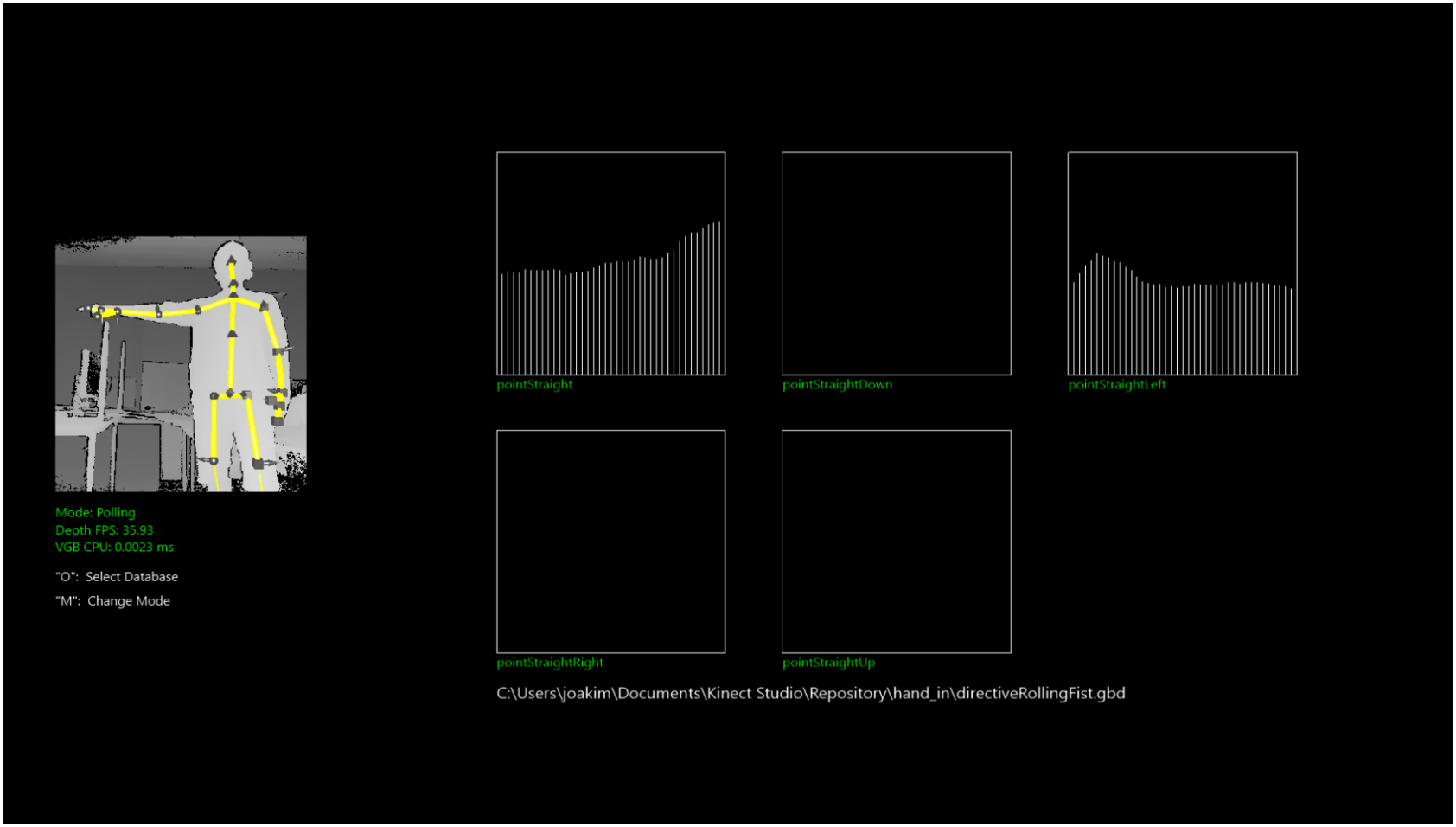
Point down


Mode: Polling
Depth FPS: 28.68
VGB CPU: 0.0029 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

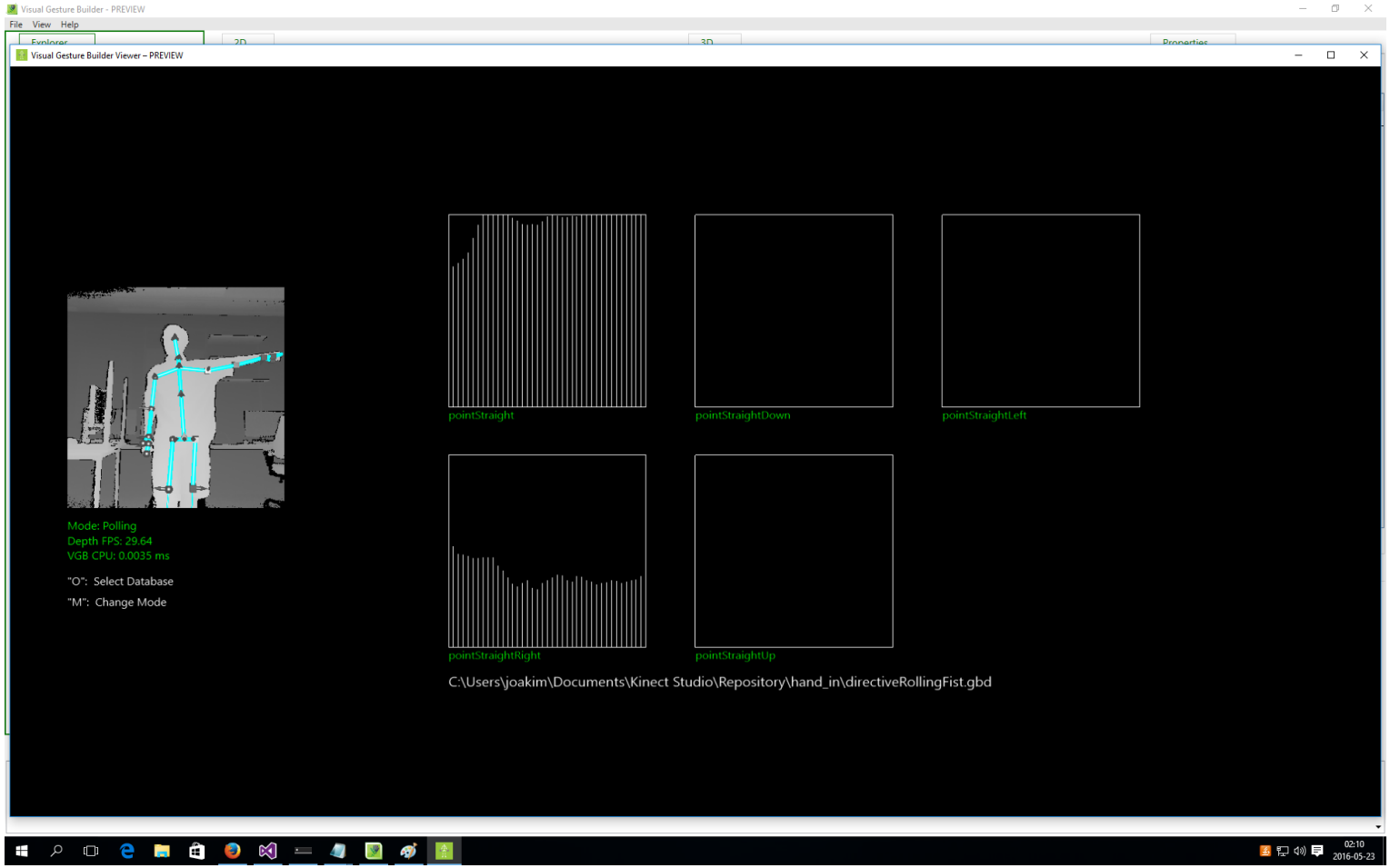
Point left


Visual Gesture Builder Viewer – PREVIEW
Mode: Polling
Depth FPS: 43.06
VGB CPU: 0.0035 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

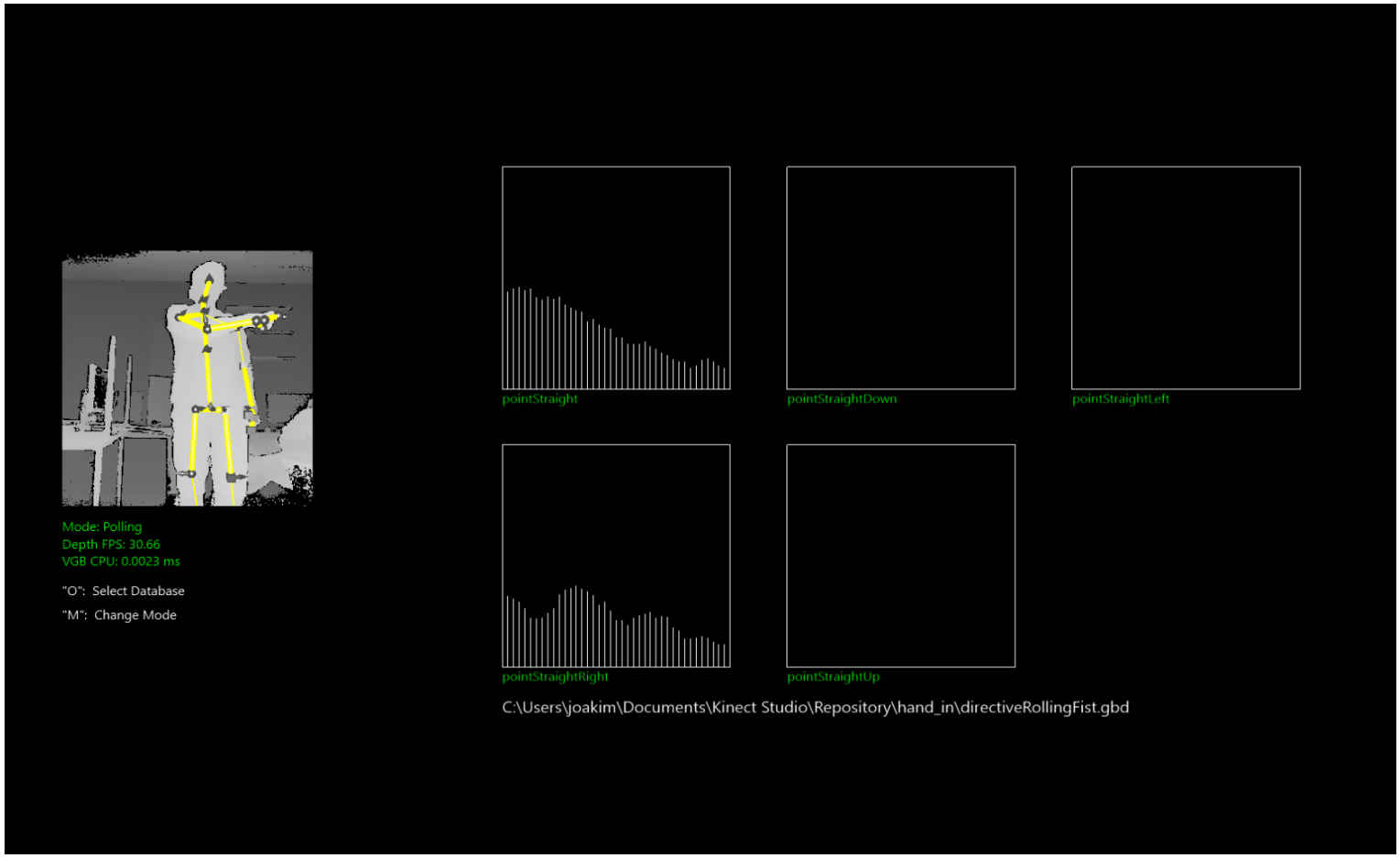
Point left


Mode: Polling
Depth FPS: 35.93
VGB CPU: 0.0023 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

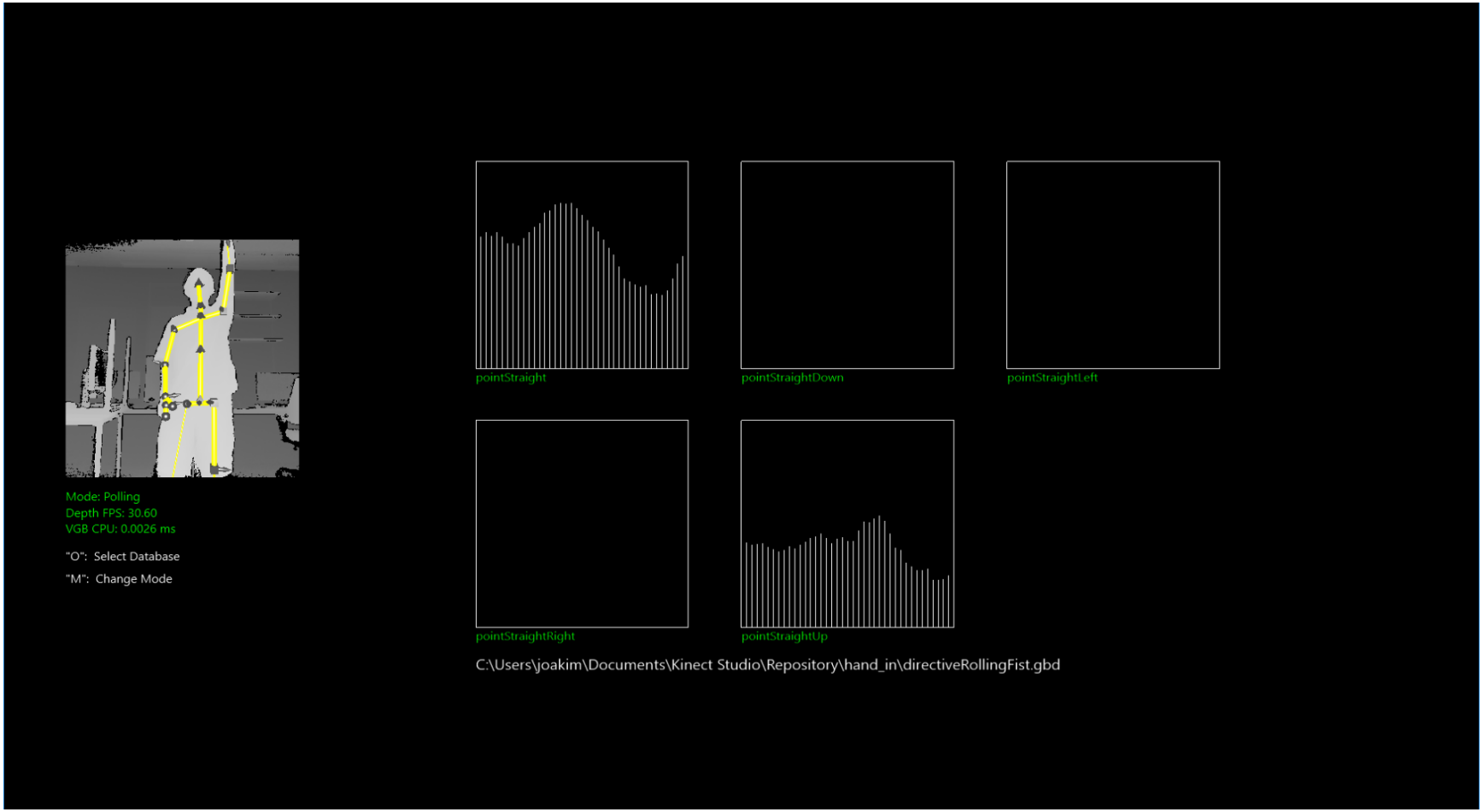
Point right


Visual Gesture Builder - PREVIEW
File View Help
Visual Gesture Builder Viewer – PREVIEW
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
Mode: Polling
Depth FPS: 29.64
VGB CPU: 0.0035 ms
"O": Select Database
"M": Change Mode
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

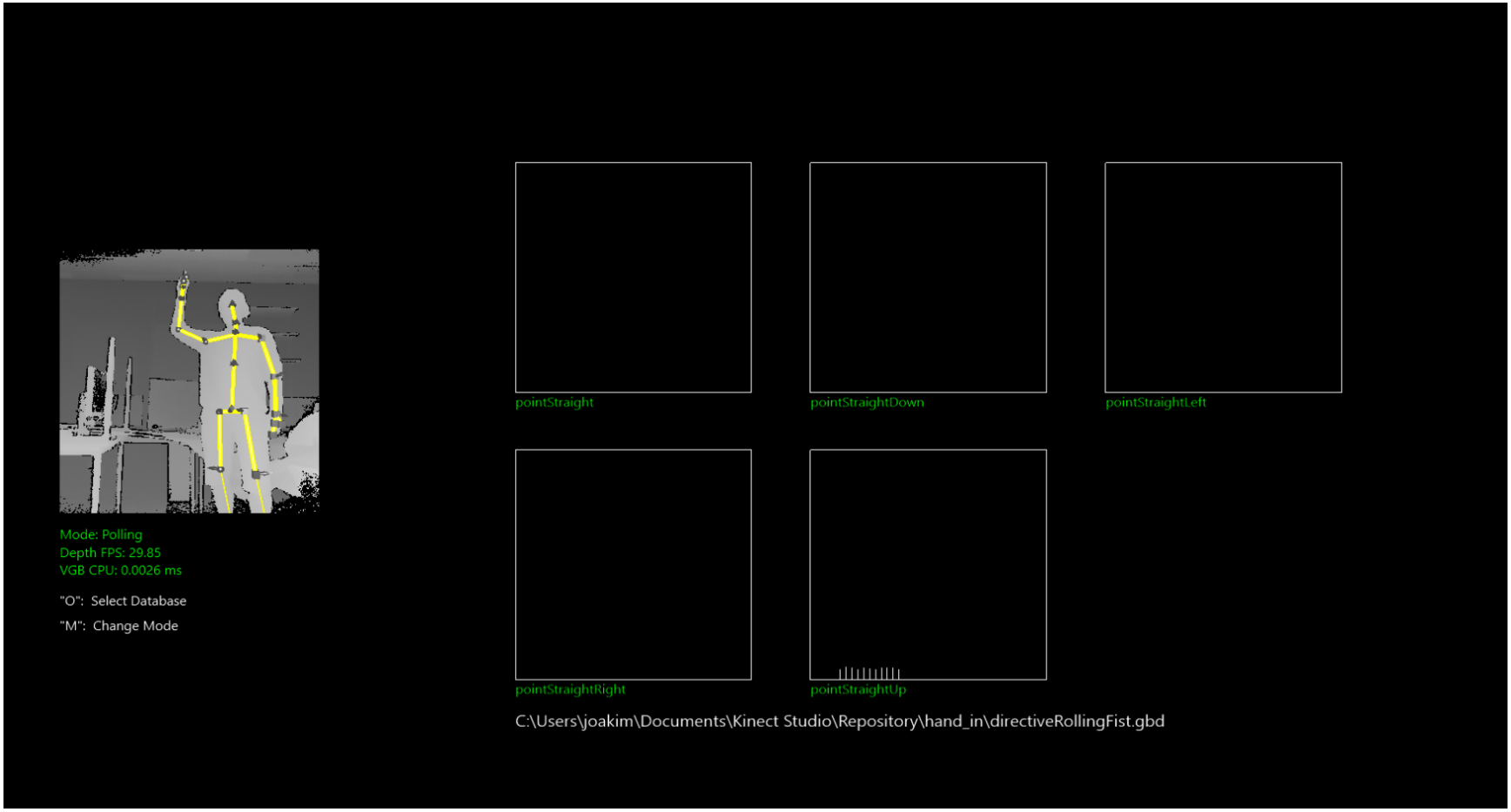
Point right


Mode: Polling
Depth FPS: 30.66
VGB CPU: 0.0023 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

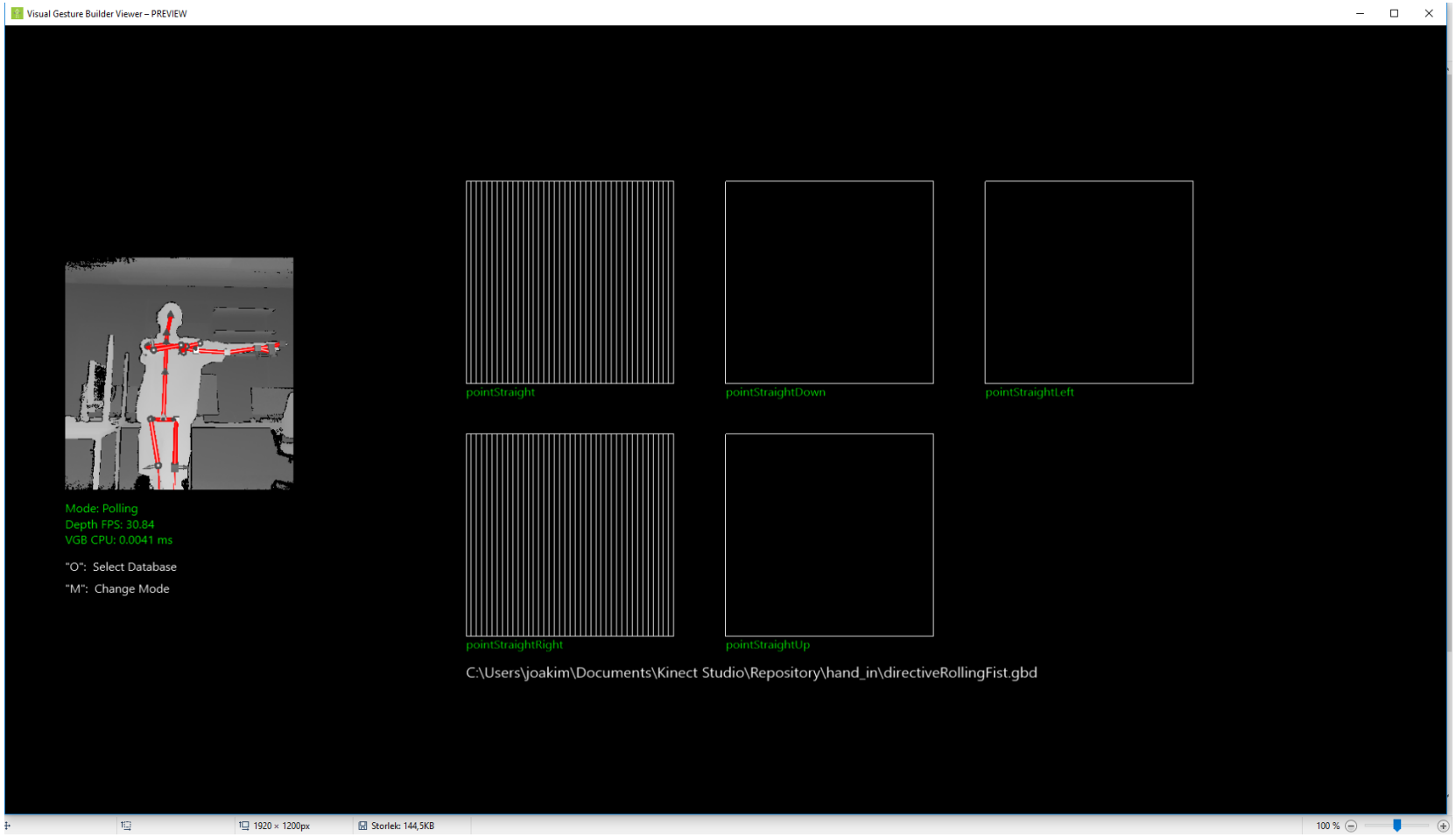
Point up


Mode: Polling
Depth FPS: 30.60
VGB CPU: 0.0026 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

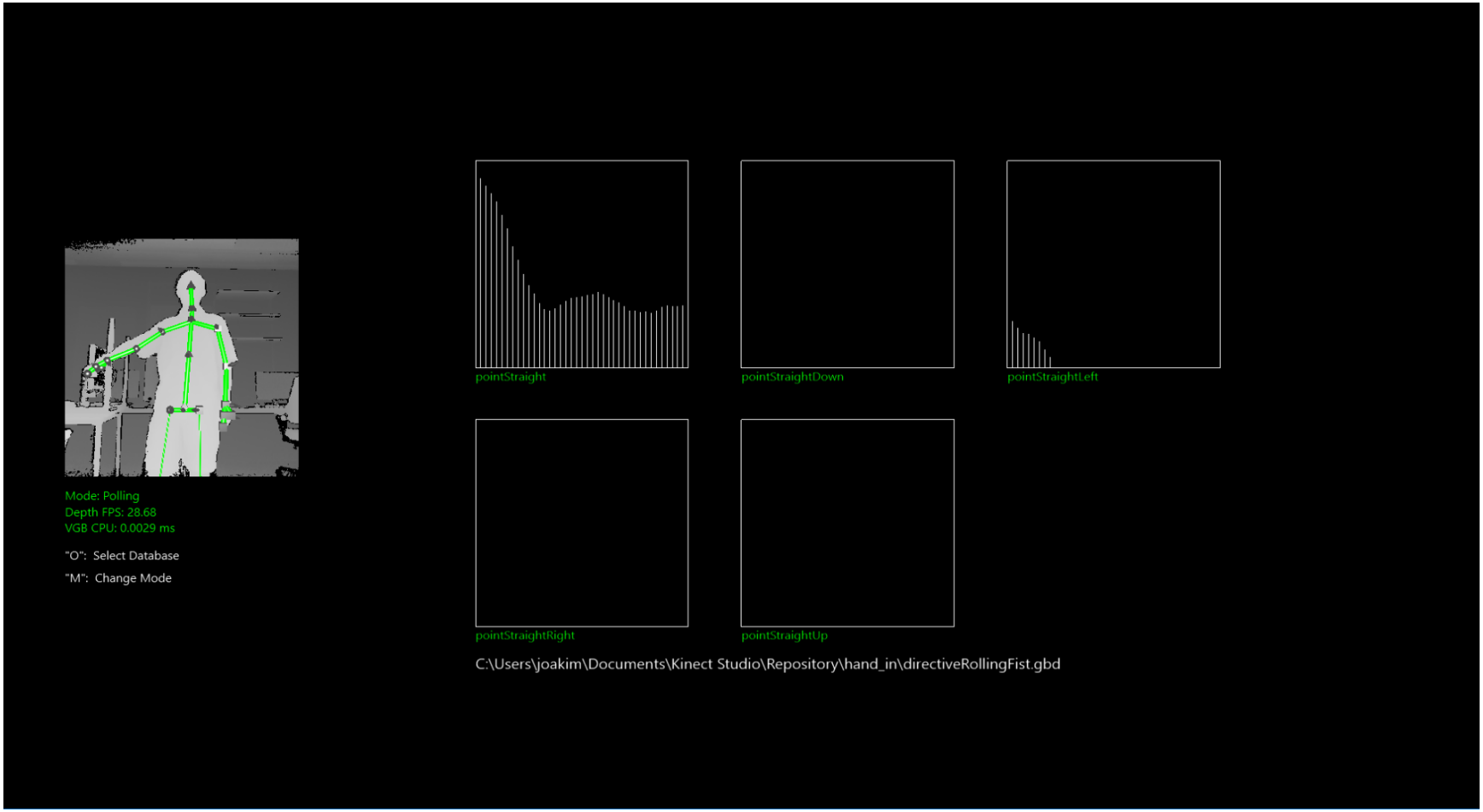
Point up


Mode: Polling
Depth FPS: 29.85
VGB CPU: 0.0026 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd

Double pointing



Finger down


Mode: Polling
Depth FPS: 28.68
VGB CPU: 0.0029 ms
"O": Select Database
"M": Change Mode
pointStraight
pointStraightDown
pointStraightLeft
pointStraightRight
pointStraightUp
C:\Users\joakim\Documents\Kinect Studio\Repository\hand_in\directiveRollingFist.gbd