1 Description of the Provided Modules

This appendix provides a short description of the modules provided with the QuantumLeap framework. If necessary, developers may write new modules to better fit their needs.

1.1 Sensors

1.1.1 Leap Motion Controller

A module that connects to the LMC and fetches frames using its JavaScript API. It returns the position of a set of articulations of the hand at one instant in time.

1.2 Filters

1.2.1 1€

A simple and efficient low-pass filter inspired by the $-family of recognizers [1]. Designed for interactive systems, it relies on an adaptive cutoff frequency that allows it to greatly reduce jitter at low speed while limiting lag at high speed.

1.3 Static and Dynamic Datasets

1.3.1 QLDynamic

A dataset composed of 17 dynamic gestures performed by four participants. Each participant performed each gesture four times above the LMC.

1.3.2 QLStatic

A dataset composed of 5 static poses performed by only one participant above the LMC.

1.4 Static Recognizers

QuantumLeap comes with two modules for static gesture recognition. Table 1 summarizes their invariance properties.

Recognizer	Type of invariance			
	Position	Scale	Rotation	Direction
$P^3+	●	●	○	●
GPSDa	●	●	●	●

Table 1: Invariance properties of the static recognizers.

1.4.1 $P^3+

A 3D version of *$P+* algorithm [2]. It is position-, direction-, and scale-invariant [3], but not rotation-invariant. It is described into more details in Appendix 1.7.

1.4.2 GPSDa

A novel position-, scale-, direction-, and rotation-invariant recognizer for hand poses regardless of the position of the hand. Its rotation-invariance makes it well-suited to situations where hand poses should be recognized regardless of their direction (*e.g.*, rotate a picture by performing a "grab" static gesture and gradually rotating the hand).

1.5 Analyzers

1.5.1 Basic Analyzer

This module extracts four values from each frame: (1) rotation, the rotation of the hand compared to the previous frame; (2) pinch, the ratio of the distances between the index and the thumb of the current

and the previous frames; (3) translation, the displacement of the palm compared to the previous frame; and (4) thumbVector, a 3D vector that gives the orientation of the thumb.

1.6 Segmenters

Three simple segmentation methods have currently been implemented to demonstrate the possibilities of QuantumLeap. We are planning on implementing more advanced segmentation techniques such as Taranta *et al.*'s Machete [4] and Chen *et al.*'s Pactolus [5]. However, support for Pactolus will only be relevant once EMG sensors are supported by QuantumLeap.

1.6.1 Zoning Segmenter

A segmenter that triggers gesture recognition when one or more selected points are within a bounding box above the sensor. After all the selected points have left, the recorded frames are sent to the recognizer.

1.6.2 Threshold Segmenter

A sensor that triggers gesture recognition when one or more parameters meet a pre-defined condition. The conditions (*e.g.*, the value of a parameter exceeds/falls below some threshold) can be manually configured from the UI. Once the conditions are no longer met, the recorded frames are sent to the recognizer.

1.6.3 Window Segmenter

A segmenter that triggers gesture recognition at regular intervals by relying on fixed-size buffers called *windows*. To adapt to varying gesture duration, it can be configured with any number of windows. Each time a new frame is received, it is pushed in the window(s), while the oldest frame is removed. Every fixed number of frames, the buffered gesture data are sent to the recognizer.

1.7 Dynamic Recognizers

QuantumLeap comes with ten modules for dynamic gesture recognition. Table 2 compares their invariance properties.

Recognizer	Type of invariance			
	Position	Scale	Rotation	Direction
P^3	●	●	○	●
Q^3	●	●	○	●
P^3+	●	●	○	●
P^3+X	●	●	○	◐
3 cent	●	●	○	○
Jackknife	●	●	○	○
$F	●	●	○	●
FreeHandUni	●	●	○	●
Rubine3D	●	●	○	○
Rubine-Sheng	●	●	○	○

Table 2: Invariance properties of the dynamic recognizers.

1.7.1 P^3

A generalization of $P [6] towards supporting 3D multi-stroke gestures which is similar to the $P3D recognizer implemented by[7]. However, unlike $P3D the recognizer $P^3 does not support 3D static poses and 2D dynamic gestures recognition. To keep memory usage and execution time low, gestures are represented as unordered sets of points, called *point clouds*. Gesture recognition happens in two phases: normalization and cloud-matching. The normalization process is similar to other $-family algorithms and is divided into three steps: (1) resample the gesture to a fixed number of equidistant points; (2) scale the gesture uniformly to keep its shape; (3) translate the centroid of the point cloud to the origin, *i.e.*, $(0, 0, 0)$. The cloud-matching phase matches a gesture's point cloud to the point cloud of each template

by associating each point from the template to exactly one point from the gesture. It then computes the resulting distance, as the sum of Euclidean distances between all pairs of matching points, and returns the template that minimizes it. This recognizer is position-, direction-, and scale-invariant, but not rotation-invariant.

1.7.2 $Q³

A 3D variant of $Q [8], which achieved a 46X speedup on average over *$P* with no loss of accuracy. It brings two key changes to $P³: (1) early abandonment of templates, as soon as the distance exceeds the current shortest distance; (2) each point cloud has a 16×16×16 3D look-up table (LUT), where each location (x, y, z) refers to the closest point. These LUTs allow $Q³ to compute a lower bound of the distance between a gesture and a template in $\mathcal{O}(n)$. If it exceeds the current shortest distance, the template can be rejected without computing the exact distance. It has the same invariance properties as $P³.

1.7.3 $P³+

A more accurate version of $P³, adapted from Vatavu's $P+ [2]. It brings three key improvements to $P³: (1) each point from the template can now be matched to more than one point of the gesture; (2) the distance between a gesture and a template takes into account the connections between consecutive points (in the form of their turning angles); (3) early abandonment of templates. This recognizer has the same invariance properties as $P³.

1.7.4 $P³+X

A variant of $P³+, which supports partial direction-invariance by keeping track of conflicting templates (*i.e.*, templates that represent the same gesture but drawn in different directions). If a gesture matches a conflicting template, its direction is compared with the direction of each conflicting template and the closest one is returned.

1.7.5 3 cent

An optimized 3D port of Wobbrock *et al.*'s $1 [9] by Caputo *et al.* [10], which recognizes uni-stroke gestures in two phases: the gesture is first normalized (similarly to *$P*) and its distance to each template is then computed as the sum of the squared distances of corresponding points. The template with the shortest distance is returned. 3¢ is position- and scale-invariant, but neither rotation- nor direction-invariant.

1.7.6 Jackknife

A general-purpose 3D gesture recognizer [11] that supports any number of articulations. Unlike most $-family recognizers, it represents gestures as time-series. It uses the nearest neighbor approach, where it compares a gesture to each template and returns the closest one. It uses *Dynamic Time Warping* as a distance measure but applies a series of correction factors to take into account differences in gesture scale and span. As a result, this recognizer is both position- and scale-invariant, but neither direction- nor rotation-invariant.

1.7.7 $F

A new recognizer that adds $P+'s flexible cloud matching [2] to $P³. As for most $-recognizers, the points of the candidate and the templates are resampled to equidistantly-spaced points, scaled within a unit box (isometric [12]), and translated so their centroid is at the origin $(0, 0, 0)$. The template with the lowest dissimilarity score is considered as the best matching template for the candidate. As opposed to $P³, $F's cloud matching is flexible, as points can be matched to more than one point. It consists of matching the points from the first cloud with their closest point from the second cloud, then matching the points from the second cloud that have not been matched yet with their closest point in the first cloud.

1.7.8 FreeHandUni

A recognizer derived from Vatavu *et al.*'s Free-Hand recognizer [13], which extends $P [6] to support 3D hand gestures. It replaces the hand pose structure with a 3D point structure (x, y, z). With this modification, FreeHandUni improves $\$P^3$ using a flexible cloud matching based on a one-to-many alignment between points [2]. The pre-processing stays the same but the matching process is more flexible: each point of the template cloud is matched with the closest point from the candidate cloud, then the remaining points from the candidate cloud are matched with the closest point from the template cloud. It returns the gesture class with the lowest dissimilarity score. FreeHandUni is different from $F in that the early abandoning is not implemented, to align the computational complexity to $\$P^3$.

1.7.9 Rubine3D

Inspired by the iGesture framework [14], Rubine3D is a feature-based recognizer that combines a set of three individual 2D Rubine recognizers [15], one for each plane XY, YZ, and ZX. Before it can recognize 3D trajectories, Rubine3D pre-processes the training templates to compute weights for each feature of each gesture class. Then, it can recognize gestures in four steps: (1) it pre-processes the gesture by scaling and filtering its points; (2) it projects it onto each plane and extracts a feature vector $(f_1, .., f_{13})$ from each of the three projections; (3) for each projection, it selects the gesture class that maximizes the similarity score (computed by combining the weights of the gesture class with the feature vector of the gesture); (4) it determines the resulting class by merging the results from the three projections.

1.7.10 Rubine-Sheng

Rubine-Sheng (RS) is a 3D recognizer inspired by Rubine. It supports 3D gestures by adding three new features proposed in the AdaBoost recognizer [16] to Rubine's existing 13 features. Aside from this difference, it is extremely similar to its 2D counterpart.

Note The implemented multipath recognizers include an optimization which avoids comparing the candidate gesture performed with one hand against the gestures performed with both hands in the training set.

References

[1] Géry Casiez, Nicolas Roussel, and Daniel Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 2527–2530, New York, NY, USA, May 2012. Association for Computing Machinery.

[2] Radu-Daniel Vatavu. Improving gesture recognition accuracy on touch screens for users with low vision. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 4667–4679, New York, NY, USA, 2017. ACM.

[3] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The Design of a GUI Paradigm Based on Tablets, Two-hands, and Transparency. In *Proceedings of the ACM International Conference on Human Factors in Computing Systems*, CHI '97, pages 35–42, New York, NY, USA, 1997. ACM.

[4] Eugene M. Taranta II, Corey R. Pittman, Mehran Maghoumi, Mykola Maslych, Yasmine M. Moolenaar, and Joseph J. Laviola Jr. Machete: Easy, efficient, and precise continuous custom gesture segmentation. *ACM Trans. Comput.-Hum. Interact.*, 28(1), January 2021.

[5] Yineng Chen, Xiaojun Su, Feng Tian, Jin Huang, Xiaolong (Luke) Zhang, Guozhong Dai, and Hongan Wang. Pactolus: A method for mid-air gesture segmentation within emg. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, page 1760–1765, New York, NY, USA, 2016. Association for Computing Machinery.

[6] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. Gestures as point clouds: A $p recognizer for user interface prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pages 273–280, New York, NY, USA, 2012. ACM.

[7] Harisson Cook, Quang Vinh Nguyen, Simeone Simoff, and Mao Lin Huang. Enabling gesture interaction with 3D point cloud. In *Proceedings of the 24th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, volume 2602, page 59–68. Computer Science Research Notes CSRN, June 2016.

[8] Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. $q: A super-quick, articulation-invariant stroke-gesture recognizer for low-resource devices. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '18, pages 23:1–23:12, New York, NY, USA, 2018. ACM.

[9] Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, pages 159–168, New York, NY, USA, 2007. ACM.

[10] F. M. Caputo, P. Prebianca, A. Carcangiu, L. D. Spano, and A. Giachetti. A 3 Cent Recognizer: Simple and Effective Retrieval and Classification of Mid-Air Gestures from Single 3D Traces. In *Proceedings of the Conference on Smart Tools and Applications in Computer Graphics*, STAG '17, page 9–15, Goslar, DEU, 2017. Eurographics Association.

[11] Eugene M. Taranta II, Amirreza Samiei, Mehran Maghoumi, Pooya Khaloo, Corey R. Pittman, and Joseph J. LaViola Jr. Jackknife: A reliable recognizer with few samples and many modalities. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 5850–5861, New York, NY, USA, 2017. ACM.

[12] Jean Vanderdonckt, Paolo Roselli, and Jorge Luis Pérez-Medina. !!ftl, an articulation-invariant stroke gesture recognizer with controllable position, scale, and rotation invariances. In *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, ICMI '18, page 125–134, New York, NY, USA, 2018. Association for Computing Machinery.

[13] Elena-Gina Craciun, Ionela Rusu, and Radu-Daniel Vatavu. Free-Hand Gesture Recognizer Pseudocode. `http://www.eed.usv.ro/mintviz/projects/GIVISIMP/data/Pseudocode2.pdf`, 2016. [Online; accessed 09-August-2020].

[14] Beat Signer, U. Kurmann, and Moira C. Norrie. igesture: A general gesture recognition framework. In *9th International Conference on Document Analysis and Recognition (ICDAR 2007), 23-26 September, Curitiba, Paraná, Brazil*, pages 954–958. IEEE Computer Society, 2007.

[15] Dean Rubine. Specifying gestures by example. In *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '91, pages 329–337, New York, NY, USA, 1991. ACM.

[16] Jia Sheng. A study of adaboost in 3D gesture recognition. technical report CSC2515, Department of Computer Science, University of Toronto, 2004.