CSCI473/573 Human-Centered Robotics
Project 2: Skeleton-Based Representations for Robot Understanding of Human Behaviors

Project Assigned: February 22, 2018
Multiple due dates (all project deliverables must be submitted via Canvas)
Deliverable 1 (Representation Code) due: March 10, 23:59:59
Deliverable 2 (Complete Code) due: March 17, 23:59:59
Deliverable 3 (Project Report) due: March, 20, 23:59:59

In this project, students will implement several skeleton-based representations (Deliverable 1) and use Support Vector Machines (SVMs) (Deliverable 2) to classify human behaviors using a public activity dataset collected from a Kinect V1 sensor. Additionally, students are required to write a report following the format of standard IEEE robotics conferences using LaTeX as Deliverable 3.

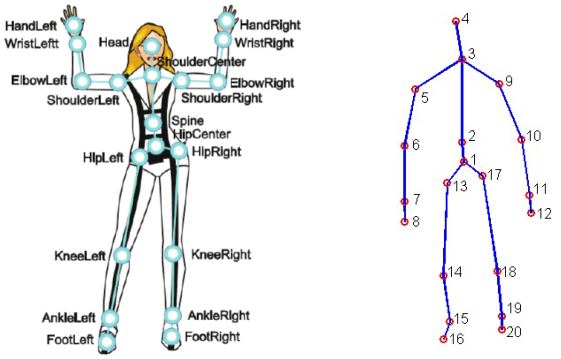
Students are required to program this project using C++ or Python in a Linux system, although students are not required to implement the project in ROS. Your code must be able to run on Ubuntu 14.04 LTS or Ubuntu 16.04 LTS for grading purpose. Project 2 must be done *individually*, although group discussion is strongly encouraged. *The deadlines are hard, and will not be extended.*

I. Dataset

The MSR Daily Activity 3D dataset[1] will be used in this projet, which is one of the most widely applied benchmark dataset in human behavior understanding tasks. This dataset contains 16 human activities, as demonstrated in Figure 2, performed by 10 human subjects. Each subject performs each activity twice, once in a standing position, and the other in a sitting position. Although this dataset contains both color-depth and skeleton data, in this project, we will only explore the skeletal information to construct **skeleton-based human representations** (Deliverable 1).

The skeleton in each frame contains 20 joints, as illustrated by Figure 1. The correspondence between joint names and joint indices is also presented in the figure. For example, Joint #1 is HipCenter, Joint #18 is KeeRight, etc.

In this project, a pre-formatted skeleton data will be used, which can be downloaded in the course website. `http://inside.mines.edu/~hzhang/Courses/CSCI473-573/assignment.html`.



(a) Names (b) Indices

Fig. 1: Skeleton joint names and indices from Kinect SDK.

This dataset in this project is a subset of original dataset, which only contains six (6) activity categories:

- CheerUp (`a08`)
- TossPaper (`a10`)
- LieOnSofa (`a12`)
- Walk (`a13`)
- StandUp (`a15`)
- SitDown (`a16`)

It is also noteworthy that this dataset is not formatted to the LIBSVM format but much easier to process then the original data format. Conversion of the data to the SVM format will be a task for Deliverable 2.

In particular, the dataset contains two folders: `Train` and `Test`. Data instances in the directory `Train` is used for training (and validation). Data instances in `Test` is used for testing the performance. Each instance has a filename like: `a12_s08_e02_skeleton_proj.txt`. This filename means the data instance belongs to activity category 12 (i.e., `a12`, that is "lie down on sofa" as in Figure 2), from human subject 8 (`s08`) at his/her second trial (i.e., `e02`). The dataset contains 16 activity categories, 10 subjects, and 2 trials each subject. Instances from subjects 1–6 are used for training (in the directory `Train`), and instances from subjects 7–10 are

If you have any questions or comments, please contact the instructor Prof. Hao Zhang at `hzhang@mines.edu`, or the TA Peng Gao at `gaopeng@mymail.mines.edu`.

This write-up is prepared using LaTeX.

[1]The MSR Daily Activity 3D dataset is removed from the author's website and no longer publicly available in 2017.

used for testing (`Test`).

When you open a data instance file, say `a12_s08_e02_skeleton_proj.txt`, you will see something like:

```
1    1    0.326    -0.101    2.111
1    2    0.325    -0.058    2.152
1    3    0.319     0.194    2.166
.
.
.
```

So each *row* contains five values, representing:

1) `frame_id`,
2) `joint_id`,
3) `joint_position_x`,
4) `joint_position_y`,
5) `joint_position_z`.

Each frame contains 20 rows that contain information of all joints in the frame.

II. CSCI 473: Deliverable 1 (Representation Construction)

Students in CSCI 473 must implement two skeleton-based representations during the Deliverable 1.

A. Relative Distances and Angles of Star Skeleton

Students in CSCI 473 are required to implement the human representation based on the **Relative Angles and Distances (RAD) of star skeleton**, as described by Algorithm 1. The objective is to implement the RAD representation to convert all data instances in the folder `Train` into a single training file `rad_d1`, each line corresponding the RAD representation of a data instance. Similarly, all instances in the folder `Test` needs to be converted into a single testing file `rad_d1.t`.

B. Customized Representations

Implement one additional skeleton-based representation other than, and different from RAD. For example, you can implement the representation of *Histogram of Joint Position Differences* (HJPD) or Histogram of Oriented Displacements (HOD), as discussed in the lecture slides, or in Section III-B for HJPD and Section III-C for HOD. A RAD representation using different joints is NOT considered as a different representation. Your code is required to output a single training file `cust_d1` for all training instances, with each row containing the customized representation of an instance, and a single testing file `cust_d1.t`, similar to the task in Section VI-A.

C. What to Submit

For the Deliverable 1, CSCI 473 students are required to submit a **single tarball**, named *T1_firstname_lastname.tar* (or .tar.gz) to the Canvas portal named P2-T1, which must contain the following items:

- A README that provides sufficient instructions needed to compile and execute your code. Your README also needs to document your implementation information, including which joints are used in the RAD representation, how the histograms are computed, and how many bins are used.

Algorithm 1: RAD representation using star skeletons

Input : Training set `Train` or testing set `Test`
Output : `rad_d1` or `rad_d1.t`

for *each instance in* `Train` *or* `Test` **do**
  for *frame t = 1, ..., T* **do**
    Select joints that form a star skeleton (Figure 3);
    Compute and store distances between body extremities to body center $(d_1^t, ..., d_5^t)$;
    Compute and store angles between two adjacent body extremities $(\theta_1^t, ..., \theta_5^t)$;
  end
  Compute a histogram of N bins for each $\boldsymbol{d}_i = \{d_i^t\}_{t=1}^T$, $i = 1, ..., 5$;
  Compute a histogram of M bins for each $\boldsymbol{\theta}_i = \{\theta_i^t\}_{t=1}^T$, $i = 1, ..., 5$;
  Normalize the histograms by dividing T to compensate for different number of frames in a data instance;
  Concatenate all normalized histograms into a one-dimensional vector of length $5(M + N)$;
  Convert the feature vector as a single line in the `rad_d1` or `rad_d1.t` file.
end
return *`rad_d1` or `rad_d1.t`*

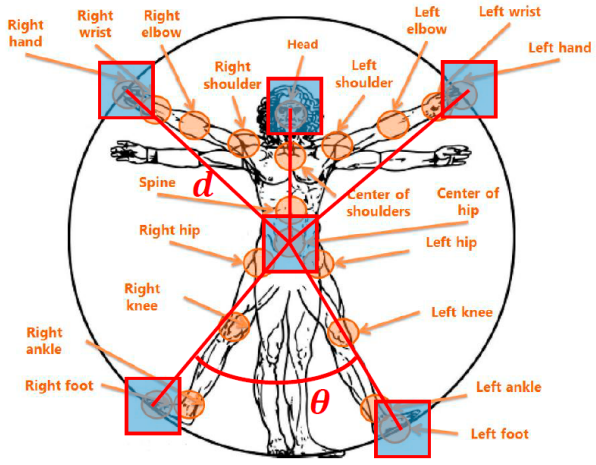


Fig. 3: Illustration of human representation based on relative distance and angles of star skeleton

- Your code to construct the RAD and customized representations.
- The generated representation data, including `rad_d1`, `rad_d1.t`, `cust_d1`, and `cust_d1.t`.

Students are allowed to include a local copy of the training and testing sets within the code directory to make your code self-contained.

III. CSCI 573: Deliverable 1 (Representation Construction)

CSCI 573 students are required to implement three specific skeleton-based representations for the Deliverable 1, including RAD, HJPD, and HOD.

A. Relative Distances and Angles of Star Skeleton

Students in CSCI 573 are required to implement the human representation based on the **Relative Angles and Distances**

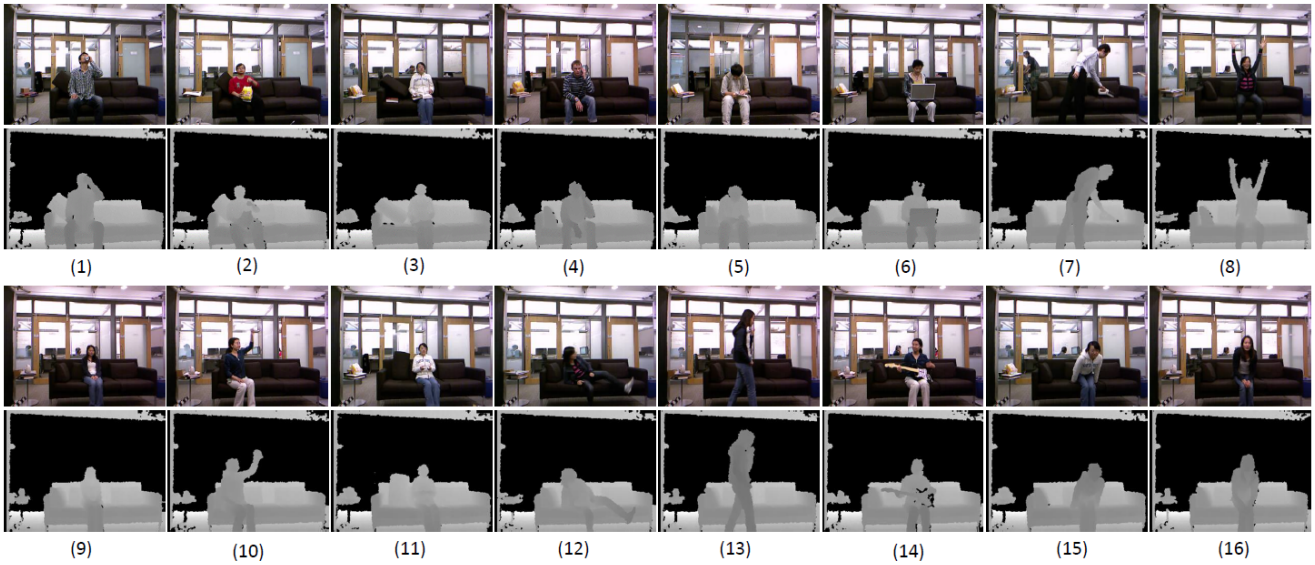


Fig. 2: The full MSR Daily Activity 3D dataset contains sixteen human activities: (1) drink, (2) eat, (3) read book, (4) call cellphone, (5) write on a paper, (6) use laptop, (7) use vacuum cleaner, (8) cheer up, (9) sit still, (10) toss paper, (11) play game, (12) lie down on sofa, (13) walk, (14) play guitar, (15) stand up, (16) sit down.

(RAD) of star skeleton, as described by Algorithm 1. The objective is to implement the RAD representation to convert all data instances in the folder `Train` into a single training file `rad_d1`, each line corresponding the RAD representation of a data instance. Similarly, all instances in the folder `Test` needs to be converted into a single testing file `rad_d1.t`. This required representation is the same as Section VI-A.

B. Histogram of Joint Position Differences (HJPD)

Given the 3D location of a joint (x, y, z) and a reference joint (x_c, y_c, z_c) in the world coordinate, the joint displacement is defined as:

$$(\Delta x, \Delta y, \Delta z) = (x, y, z) - (x_c, y_c, z_c) \quad (1)$$

The reference joint can be the skeleton centroid or a fixed joint. For each temporal sequence of human skeletons (in a data instance), a histogram is computed for the displacement along each dimension, i.e., $\Delta x, \Delta y, \Delta z$. Then, the computed histograms are concatenated into a single vector as a feature.

This HJPD representation is similar to the RAD representation, except that it uses all joints and ignores the pairwise angles. Refer to Section 3.3 of reference [1] for more details, which is available online at:

`http://staffhome.ecm.uwa.edu.au/~00053650/papers/hossein_WACV2014.pdf`.

Your code is required to generate a single training file `hjpd_d1` for all training instances, with each row containing the HJPD representation of an instance, and a single testing file `hjpd_d1.t`.

C. Histogram of Oriented Displacements (HOD)

You need to implement the skeleton-based representation of Histogram of Oriented Displacements (HOD), as introduced in Section 3 of reference [2], including the Temporal Pyramid. The paper is available online at:

`http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6967`.

Your code is required to generate a single training file `hod_d1` for all training instances, with each row containing the HOD representation of an instance, and a single testing file `hod_d1.t`.

D. What to Submit

For the Deliverable 1, CSCI 573 students are required to submit a **single tarball**, named *T1_firstname_lastname.tar* (or .tar.gz) to the Canvas portal named P2-T1, which must contain the following items:

- A README that provides sufficient instructions needed to compile and execute your code. Your README also needs to document your implementation information, for example, including which joints are used in the RAD representation, and how the histograms are computed and how many bins are used in your HJPD and HOD representations.
- All your code to construct the RAD, HJPD, and HOD representations.
- All the generated skeleton-based representation data, including `rad_d1`, `rad_d1.t`, `hjpd_d1`, `hjpd_d1.t`, `hod_d1`, and `hod_d1.t`.

Students are allowed to include a local copy of the training and testing sets within the code directory to make your code-self contained.

IV. Support Vector Machines (Part of Deliverable 2)

The second objective of Project 2 (in Deliverable 2) is to understand how to apply SVMs to enable robot learning in practical applications (i.e., activity understanding in our case).

We will apply LIBSVM as our learning algorithm, which is an excellent open source implementation of SVMs developed by Chang and Lin [3]. The LIBSVM library provides software support for a variety of SVMs, with the source code available in C++, which also provides an interface to Python and many other programming languages and environments. Here's the link to the LIBSVM webpage:

`http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Before working on Deliverable 2, you will need to install LIBSVM, get familiar with this library, and understand how to convert the output data files from your Deliverable 1 to the format required by LIBSVM, as described in the following subsections.

A. Installing LIBSVM

First, follow the instructions on the LIBSVM website for downloading the software. The most recent release is Version 3.22 (December 2016). Note that the README file within the package provides very helpful information for using the LIBSVM package. In addition, you are required to read through and make sure to have a good understanding of the **Practical Guide** provided by the authors:

`http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

B. Getting Familiar with LIBSVM

You are required to follow the examples in Appendix A of the Practical Guide to get familiar with how to use LIBSVM. The exemplary datasets (e.g., svmguide1) are available online from the LIBSVM directory, already formatted into the form expected by LIBSVM, here:

`http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

C. Input Data Format Required by LIBSVM

Now, given the histograms, e.g., the skeleton-based representations such as `rad_d1` or `rad_d1.t` you have computed in Deliverable 1, you will need to convert them into data files with the a format that can be used by LIBSVM: `rad_d2` and `rad_d2.t` for training and testing respectively. Specifically, this format of data file now becomes:

<label> <index1>:<value1> <index2>:<value2> ...
.
.
.

Each line contains an instance and is ended by the '\n' character. For classification (as in this project), <label> is an integer indicating the class label (multi-class is supported). Indices must be in *ASCENDING* order. Labels in the testing instances are only used for evaluation to calculate accuracy or errors.

Note that the histograms you computed in Deliverable 1 must be converted to this new data format for the LIBSVM software to process. After obtaining a single data file `rad_d2` from training data and a single file `rad_d2.t` from testing instances (both of which follow the LIBSVM format), you need to use LIBSVM to learn a C-SVM model with the radial basis function (RBF) kernel (see the Practical Guide) from the training data `rad_d2`. Then, you need to use the learned model to make predictions of the testing data in `rad_d2.t`. The predicted class label will be automatically generated by LIBSVM with a name `rad_d2.t.predict`. Please refer to the examples in the Practical Guide. Then, you will need to evaluate the performance (e.g., accuracy and confusion matrix) of your robot learning method in Deliverable 2, combined with the skeleton-based representations you built in Deliverable 1.

V. **CSCI 473**: DELIVERABLE 2 (ROBOT LEARNING FROM DATA)

A. Tasks to Perform

Students in CSCI 473 need to perform the following tasks to finish Deliverable 2, including:

1) Read and understand Section IV as well as the Practical Guide of LIBSVM (the theory and practical usage of LIBSVM have been discussed in the class). Go through the examples provided by the Practical Guide.
2) Convert the training and testing files (i.e., the outputs of your code to build the representations in Deliverable 1 to a format that can be used by LIBSVM. Name them as `rad_d2`, `rad_d2.t`, `cust_d2`, and `cust_d2.t`
3) Apply LIBSVM to learn a C-SVM model with the RBF kernel from the training data, and use the learned model to predict behavior labels of the testing data, which will generate a result file, e.g., `rad_d2.t.predict`, and `cust_d2.t.predict`.
4) Evaluate its performance based upon the output results using the performance metrics of *accuracy* and *confusion matrix*. Your implementation's accuracy based on each representation must be better than 50%.

B. What to Submit

For the Deliverable 2, CSCI 473 students are required to submit a **single tarball**, named *T2_firstname_lastname.tar* (or .tar.gz) to the Canvas portal named P2-T2, which must contain the following items:

- The graphs of the grid search (from grid.py) obtained in the experiments for both representations (i.e., RAD and the customized representation);
- What are the "best" values of C and γ of your C-SVMs for both representations, which need to be written in the README;
- All the converted representation files and output prediction result files;
- Any code you write in Deliverable 2 and the README (with the same requirement as in Deliverable 1).

VI. **CSCI 573**: DELIVERABLE 2 (ROBOT LEARNING FROM DATA)

A. Tasks to Perform

Students in CSCI 573 need to perform the following tasks to finish Deliverable 2, including:

1) Read and understand Section IV as well as the Practical Guide of LIBSVM (the theory and practical usage of LIBSVM have been discussed in the class). Go through the examples in the Practical Guide, and understand how to integrate LIBSVM into your source code.
2) Convert the training and testing files (i.e., the outputs of your code to build the representations in Deliverable 1 to a format that can be used by LIBSVM. This must be done for all three representations (i.e., RAD, HJPD, and HOD).
3) Apply LIBSVM to learn a C-SVM model with the RBF kernel from the training data, and use the learned model to predict behavior labels of the testing data, which will generate a result file for all the representations.
4) Write an integrated program that reads data from the training and testing directories, creates a given representation (specified by a command-line flag), performs robot learning, and outputs the information of *accuracy* and *confusion matrix* to the screen. You are free to utilize any way to implement the *integrated* program, such as using the C++ API provided by the LIBSVM library. Your implementation's accuracy based on each representation must be better than 60%.
5) Analyze how the accuracy varies according to different numbers of bins.

B. What to Submit

For the Deliverable 2, CSCI 573 students are required to submit a **single tarball**, named *T2_firstname_lastname.tar* (or .tar.gz) to the Canvas portal named P2-T2, which must contain the following items:

- The graphs of the grid search (from grid.py) obtained in the experiments for all the representations (i.e., RAD, HJPD, and HOD);
- What are the "best" values of C and γ of your C-SVMs for all the representations, which can be written in the README;
- All the converted representation files and output prediction result files;
- A figure showing how accuracy varies according to the number of bins;
- The code you write in Deliverable 2 and the README (with the same requirement as in Deliverable 1).

VII. **CSCI 473**: Deliverable 3 (Project Report)

As the last deliverable of your Project 2, you must prepare a single 1-3 page document (in pdf format, using LaTeX and 2-column IEEE style that is similar to this project write-up). You may find the LaTeX template at:

`http://ras.papercept.net/conferences/support/tex.php`

Your report should discuss the design details (of both your representations and the SVM models) and the experimental results (including the accuracy and confusion matrix). The documentation you turn in must be in pdf format in a single file generated using LaTeX. Name your paper *T3_firstname_lastname.pdf* and submit it to the Canvas portal named P2-T3.

VIII. **CSCI 573**: Deliverable 3 (Project Paper)

As the last deliverable of your Project 2, you must prepare a single 4-6 page document (in pdf format, using LaTeX and 2-column IEEE style as this project write-up). You may find the LaTeX template at:

`http://ras.papercept.net/conferences/support/tex.php`

Your project paper should includes the following:

- An abstract containing 200 to 300 words to summarize your findings.
- A brief introduction describing the skeleton-based representations and SVM models, as well as your implementations.
- A detailed description of your experiments, with enough information that would enable someone to recreate your experiments.
- An explanation of the results. Include all the experimental details and results. Use figures, graphs, and tables where appropriate. Your results should make it clear that the software has in fact worked. Accuracy and confusion matrix must be presented and discussed. In particular, the accuracy of your approach must be better than 50%. The graphs showing the changes of accuracy according to different numbers of bins must also be included and discussed.
- A discussion of the significance of the results.

The reader of your paper must be able to understand what you have done and what your software does without looking at the code itself.

The documentation you turn in must be in pdf format in a single file. Name your paper *T3_firstname_lastname.pdf* and submit it to the Canvas portal named P2-T3.

IX. Grading

The total score of Project 2 is 10 points. Your grade will be based on the quality of your project implementation and the documentation of your findings in the report.

- 30%: The quality of Deliverable 1. You should have a "working" software implementation, which means that your skeleton-based representations are implemented in software, your code runs without crashing, and performs the behavior understanding task.
- 45%: The quality of Deliverable 2 (with almost the same requirements to the Deliverable 1).
- 25%: The quality of Deliverable 3, that is – the project report prepared in LaTeX using IEEE styling and submitted in the pdf format – Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure or graph illustrates. The content should include all the experimental results and discussions mentioned previously in this document.

Students in CSCI573 will be graded more strictly on the quality of the code implementation and paper presentation. The instructor expects a more through analysis of the experimental results, and a good implementation of skeleton-based representations and robot learning methods for human behavior understanding. The paper should have the "look and feel" of a technical conference paper, with logical flow, good grammar, sound arguments, illustrative figures, etc.

REFERENCES

[1] H. Rahmani, A. Mahmood, D. Q. Huynh, and A. Mian, "Real time action recognition using histograms of depth gradients and random decision forests," in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.

[2] M. A. Gowayyed, M. Torki, M. E. Hussein, and M. El-Saban, "Histogram of oriented displacements (hod): Describing trajectories of human joints for action recognition.," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[3] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.