User Manual for PARSED (PARticle SEgmentation Detector)

(Version 1.0)

1 Installation Requirements

1.1 Hardware

- A desktop computer or workstation with multi-core CPUs and Linux OS;
- An NVIDIA CUDA compatible GPU with *Kepler* architecture or newer (SM ≥ 3.5);
- A fast storage disk with enough I/O speed. We recommend solid state drive (SSD) for high processing speed.

1.2 Software dependency

All required software and packages are listed in Table 1.

Table 1: Software and packages required for running PARSED

Software	Description	Version
python3	Python interpreter version 3	3.6
CUDA	NVIDIA `CUDA` acceleration library	8.0
docker	Container platform for software deployment	1.10
nvidia-docker	NVIDIA container runtime for docker	2.0
h5py	Python wrapper for HDF5 data format	2.7.1
Keras	`Tensorflow` API	2.0.8
mrcfile	Python wrapper for MRC file format	1.0.1
pandas	Data frame module	0.20.3
trackpy	A python program of *Crocker-Grier* algorithm	0.4.1
tensorflow-gpu	GPU accelerated `TensorFlow`	1.3.0
numba	Python JIT acceleration for `trackpy`	0.37.0
matplotlib	Python 2D plotting library	2.1.0
cv2	`OpenCV` library for morphological interfaces	3.3.1

Specific notes:

- `python3` interpreter ≥ 3.6. For a system-wide build of `PARSED`, please refer documents from other Linux distributions (e.g., CentOS 7) to gain python3 support;
- `Keras` ≥ 2.0;
- NVIDIA `CUDA` library ≥ 8.0;
- `docker` platform ≥ 1.10 (containerized installation, see details in Subsec. 2.3), `nvidia-docker` (container runtime for NVIDIA `CUDA` capacity).

2 Software Deployments

We provide two deployment methods:

- System-wide deployment;
- Containerized deployment.

We recommend the second one to avoid dependency problems between different systems. To enable

CUDA functionality, both methods need following *basic system environment* with NVIDIA proprietary driver.

2.1 Basic system configuration

(1) Install NVIDIA proprietary driver

Follow NVIDIA official instructions at https://www.nvidia.com/Download/index.aspx?lang=en.

(2) Check NVIDIA driver

Run `nvidia-smi` in your shell:

```
$ nvidia-smi
```

2.2 System-wide deployment

(1) Install CUDA libraries

Follow NVIDIA official instructions at https://developer.nvidia.com/cuda-downloads.

(2) Install required packages/software

Use `pip3` to fetch python packages and install `pip3` with Linux package manager such as `apt-get`:

```
$ sudo apt-get install pip3
```

Then, install python packages:

```
$ sudo pip install h5py==2.7.1 \
  Keras==2.0.8 \
  mrcfile==1.0.1 \
  numba==0.37.0 \
  pandas==0.20.3 \
  trackpy==0.4.1 \
  matplotlib==2.1.0
```

For TensorFlow and OpenCV, please refer their official documents for complete installation.

- TensorFlow at https://www.tensorflow.org/.
- OpenCV at https://opencv.org/.

(3) Download our compressed file `parsed_v1.tar.gz` and extract related files, as listed Table 2:

Table 2: Files in `parsed_v1.zip`

`Dockerfile`	Instruction file for installing `docker`
`parsed_main.py`	PARSED main program
`pre_train_model.h5`	Pre-trained model of PARSED neural networks
`mic_preprocess.py`	Module for pre-processing raw micrographs
`particle_mass.py`	Program to generate particle mass distribution

2.3 Containerized deployment

(1) Install `docker`

Follow official documents and instructions at https://www.docker.com/get-started.

Install `docker` with Linux package manager:

```
$ sudo apt-get install docker.io
```

(2) Install NVIDIA containerized runtime `nvidia-docker`
Follow official `nvidia-docker` wiki page at *https://github.com/NVIDIA/nvidia-docker*.

(3) Build PARSED locally by downloading our compressed file `parsed_v1.tar.gz`, and extracting related files, as listed in Table 2. Make sure current user is in `docker` group and then build the `PARSED` docker image:

```
$ docker build -t parsed:local .
```

2.4 Installation check

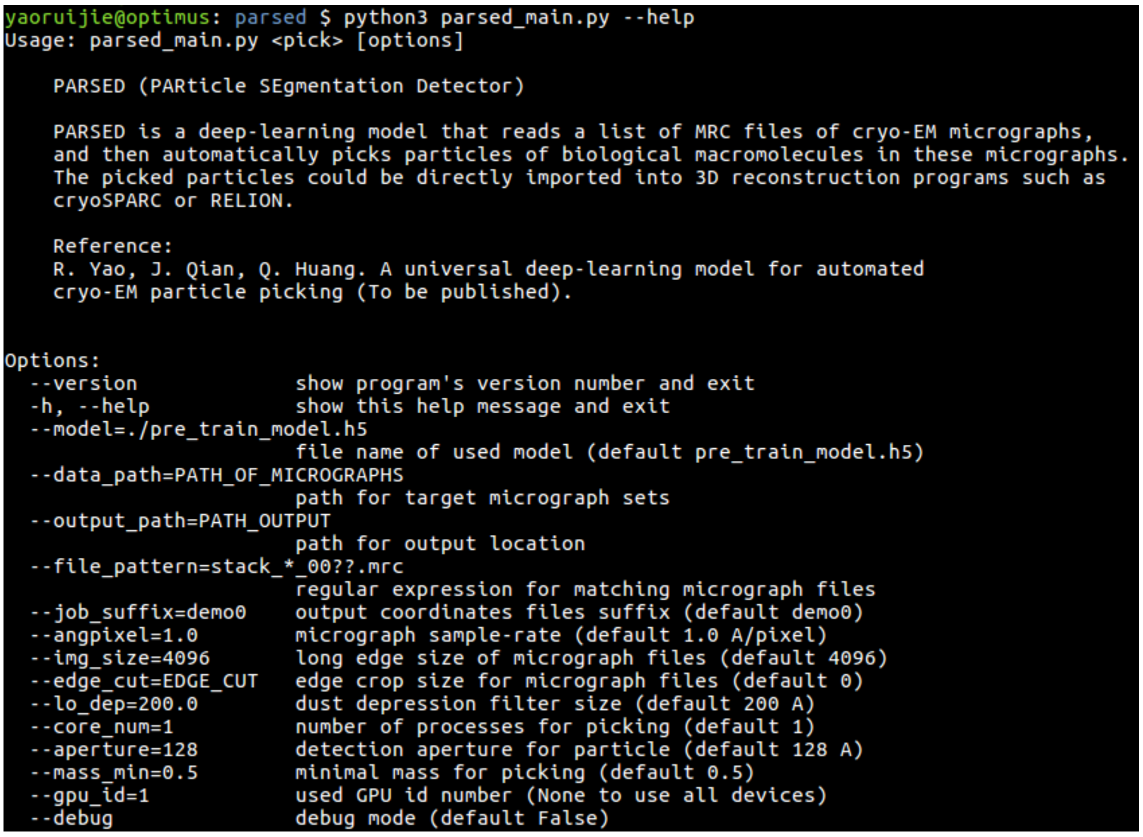
To check system-wide deployment, run `PARSED` as:

```
$ python3 parsed_main.py –help
```

To check containerized deployment, please run:

```
$ docker run --runtime=nvidia -it --rm parsed:local python3
  parsed_main.py --help
```

If everything works well, a built-in help page will print on screen, as shown in Figure 1.



```
yaoruijie@optimus: parsed $ python3 parsed_main.py --help
Usage: parsed_main.py <pick> [options]

    PARSED (PARticle SEgmentation Detector)

    PARSED is a deep-learning model that reads a list of MRC files of cryo-EM micrographs,
    and then automatically picks particles of biological macromolecules in these micrographs.
    The picked particles could be directly imported into 3D reconstruction programs such as
    cryoSPARC or RELION.

    Reference:
    R. Yao, J. Qian, Q. Huang. A universal deep-learning model for automated
    cryo-EM particle picking (To be published).


Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  --model=./pre_train_model.h5
                        file name of used model (default pre_train_model.h5)
  --data_path=PATH_OF_MICROGRAPHS
                        path for target micrograph sets
  --output_path=PATH_OUTPUT
                        path for output location
  --file_pattern=stack_*_00??.mrc
                        regular expression for matching micrograph files
  --job_suffix=demo0    output coordinates files suffix (default demo0)
  --angpixel=1.0        micrograph sample-rate (default 1.0 A/pixel)
  --img_size=4096       long edge size of micrograph files (default 4096)
  --edge_cut=EDGE_CUT   edge crop size for micrograph files (default 0)
  --lo_dep=200.0        dust depression filter size (default 200 A)
  --core_num=1          number of processes for picking (default 1)
  --aperture=128        detection aperture for particle (default 128 A)
  --mass_min=0.5        minimal mass for picking (default 0.5)
  --gpu_id=1            used GPU id number (None to use all devices)
  --debug               debug mode (default False)
```

Figure 1: Help page for PARSED

3 Picking Particles with PARSED

In this section, we show how to use `PARSED` to pick particles from cryo-EM micrographs. Parameters for running `PARSED` are listed in Table 3.

Table 3: Parameters for running `PARSED`

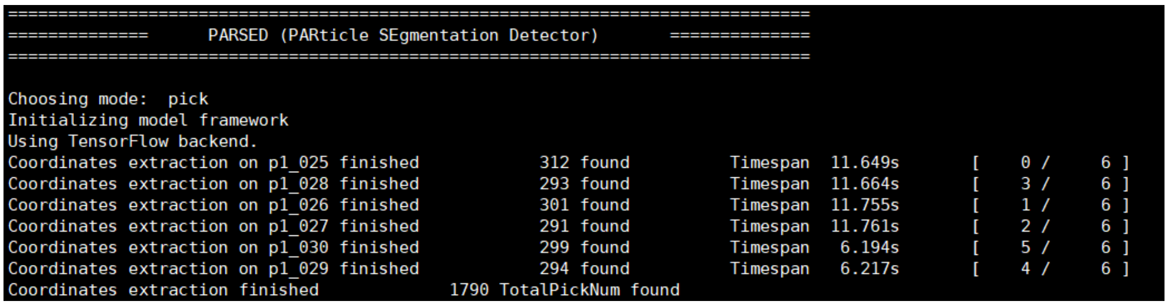
Parameters	Description
`--model`	pre-trained segmentation model
`--data_path`	directory path of input data
`--output_path`	storage path of output data
`--file_pattern`	filename of micrographs, also accept Unix bash-like regular expression (RegEx)
`--job_suffix`	suffix for output filename
`--angpixel`	sample rate for specified micrographs (in Å/pixel)
`--img_size`	size of micrographs (the largest one of Width and Height, in pixel)
`--edge_cut`	cropped edge size (in pixel)
`--core_num`	parallel running process number
`--aperture`	diameter of particle-picking aperture (in Å)
`--mass_min`	minimal picking mass of detected blobs (particle candidates)
`--gpu_id`	IDs of used GPUs (start from 0)

3.1 Running `PARSED`

With micrographs in subdirectory `demo`, run `PARSED` to detect particles in these micrographs:

```
$ python3 -W ignore parsed_main.py \
 --model=./pre_train_model.h5 --data_path=./demo \
 --output_path=./output --file_pattern=*.mrc \
 --angpixel=1.34 --img_size=4096 --edge_cut=0 \
 --job_suffix=autopick --core_num=4 -- aperture=160 \
 --mass_min=4 --gpu_id=1
```

Example of program outputs on screen is shown in Figure 2.



```
================================================================================
==============       PARSED (PARticle SEgmentation Detector)        ==============
================================================================================

Choosing mode:  pick
Initializing model framework
Using TensorFlow backend.
Coordinates extraction on p1_025 finished          312 found         Timespan  11.649s      [    0 /     6 ]
Coordinates extraction on p1_028 finished          293 found         Timespan  11.664s      [    3 /     6 ]
Coordinates extraction on p1_026 finished          301 found         Timespan  11.755s      [    1 /     6 ]
Coordinates extraction on p1_027 finished          291 found         Timespan  11.761s      [    2 /     6 ]
Coordinates extraction on p1_030 finished          299 found         Timespan   6.194s      [    5 /     6 ]
Coordinates extraction on p1_029 finished          294 found         Timespan   6.217s      [    4 /     6 ]
Coordinates extraction finished           1790 TotalPickNum found
```

Figure 2: PARSED running screenshot

The coordinates of the detected particles will be stored in `STAR` format. Suffix for filenames is given by `--job_suffix`, and location is given by `--output_path` .

3.2 Selecting detected particles

`PARSED` records the properties of detected blobs (particle candidates) in separate files with suffix `_param`, e.g., their mass (i.e., blob brightness). To reduce falsely positive picking, one could filter the detected particles by using *grep*, *sed*, and *awk*, etc. Also, we offer a python script `particle_mass.py` to generate blob mass distribution for all the detected particles, and then select some of them for further single-particle analysis (SPA) with a mass selection threshold (see also Methods).

Create the mass distribution of detected particles to `tmp_hist.jpg`:

```
$ python3 particle_mass.py drawmass --pick_output=./output \
  --job_suffix=autopick --tmp_hist=tmp_hist.jpg
```

Two typical mass distributions are illustrated in Figure 3.

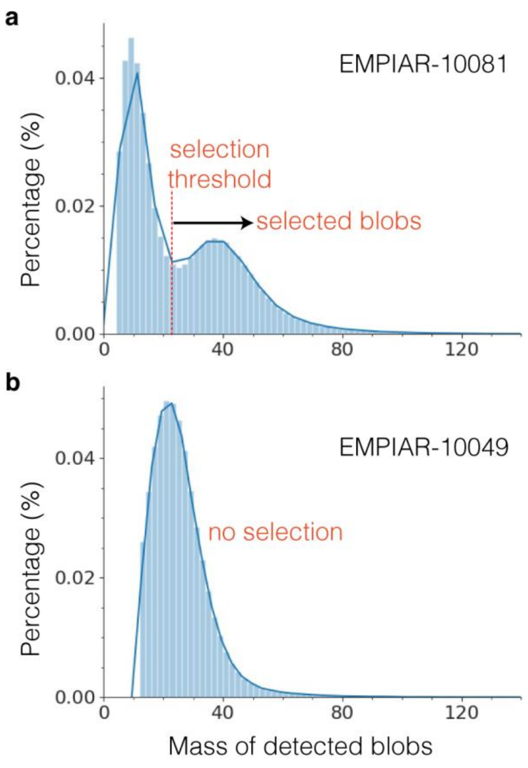


Figure 3: Mass distributions of detected blobs (particle candidates). **a**, A two-peak distribution with a selection threshold at mass of ~21 (from the dataset EMPIAR-10081). In this case, only those detected blobs with mass values greater than the selection threshold are selected as the final picked particles for further investigations. **b**, A one-peak distribution without selection (from the dataset EMPIAR-10049). In this case, all detected blobs are selected as the final picked particles.

Then, determine whether or not a selection is needed:

```
$ python3 particle_mass.py cutoff --pick_output=./output \
  --job_suffix=autopick --output_suffix=checked  --thres=28
```

Output results will be generated in the same directory.

4 Importing Picked Particles into 3D Reconstruction Program

In this section, we show how to import the final picked particles into *ab initio* 3D reconstruction program `RELION` to build 3D density maps with standard single-particle analysis (SPA) procedures. Please see more details about `RELION` at: *http://www2.mrc-lmb.cam.ac.uk/relion/index.php?title=Main_Page*.

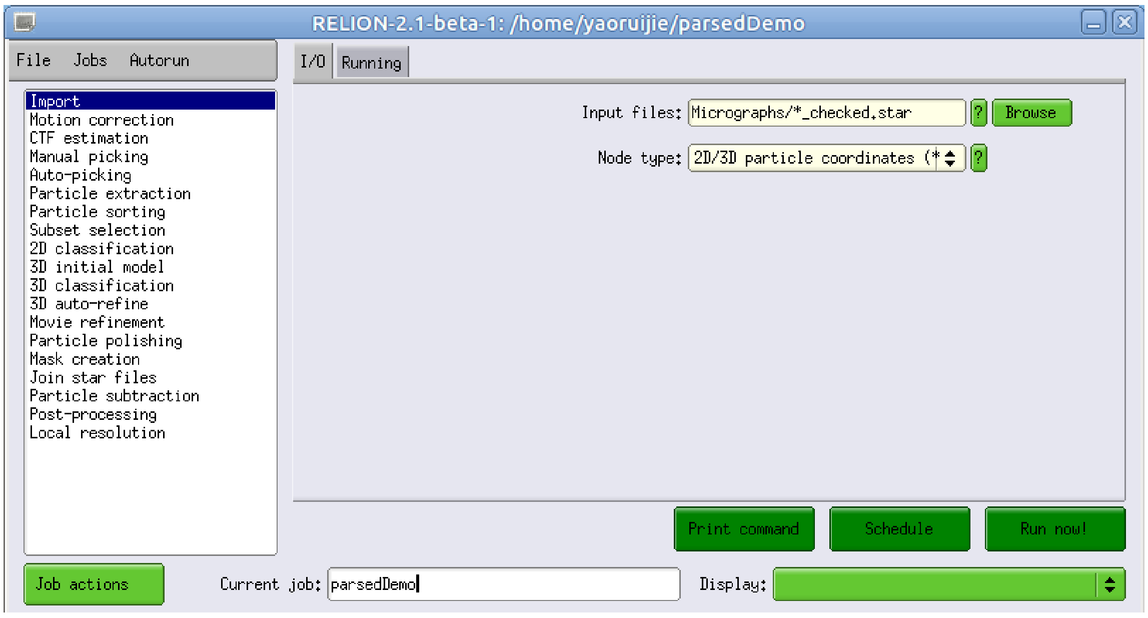
4.1 Copying particles to `RELION` work directory

To import the picked particles into `RELION`, we copy the output results back to `RELION` work directory. Suppose RELION `Import`, and `CTF estimation` for the micrographs have been completed, and work directory is: `~/Relion2Test/demodir`, directory for raw micrographs is: `~/Relion2Test/demodir/Micrographs`; then, copy the results to the micrograph folder for `RELION`:

```
demodir $ cp ~/parsed/output/*_checked.star Micrographs/
```

4.2 Importing particles into `RELION`

Start `RELION` and select `Import` option in left, as shown in Figure 4. Here, we designate `Import` job as `parsedDemo`, and suffix as `_checked`.


RELION-2.1-beta-1: /home/yaoruijie/parsedDemo

Input files: Micrographs/*_checked.star

Node type: 2D/3D particle coordinates (*

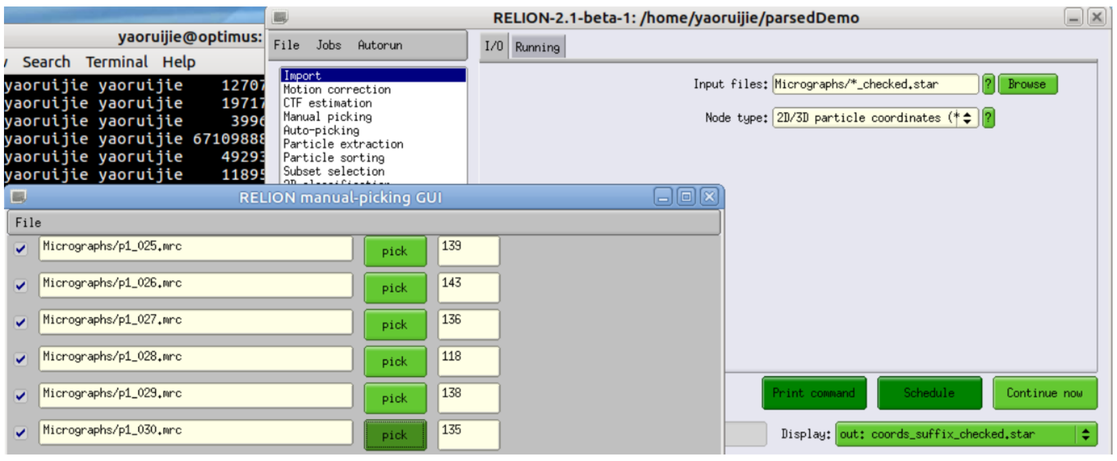
Current job: parsedDemo

Figure 4: RELION panel for `Import` job

4.3 Checking imported particles with `RELION`

Before displaying picked particles, we create a `Manual picking` job in `RELION`. Next, as illustrated in Figure 5, we select our `Import` job, click `Display` green drop-down menu and then item `out:coords_suffix_checked.star`, the picked particles will be displayed, as in Figure 6.

After importing the particles, further steps in the SPA workflow, such as particle extraction, 2D classification and 3D reconstruction etc., could be carried out with standard procedures in `RELION`.



```
yaoruijie yaoruijie    12707
yaoruijie yaoruijie    19717
yaoruijie yaoruijie     3996
yaoruijie yaoruijie 67109888
yaoruijie yaoruijie    49293
yaoruijie yaoruijie    11895
```

Input files: Micrographs/*_checked.star

Node type: 2D/3D particle coordinates (*

Display: out: coords_suffix_checked.star

✓	Micrographs/p1_025.mrc	pick	139
✓	Micrographs/p1_026.mrc	pick	143
✓	Micrographs/p1_027.mrc	pick	136
✓	Micrographs/p1_028.mrc	pick	118
✓	Micrographs/p1_029.mrc	pick	138
✓	Micrographs/p1_030.mrc	pick	135

Figure 5: `RELION` panels for checking imported particles

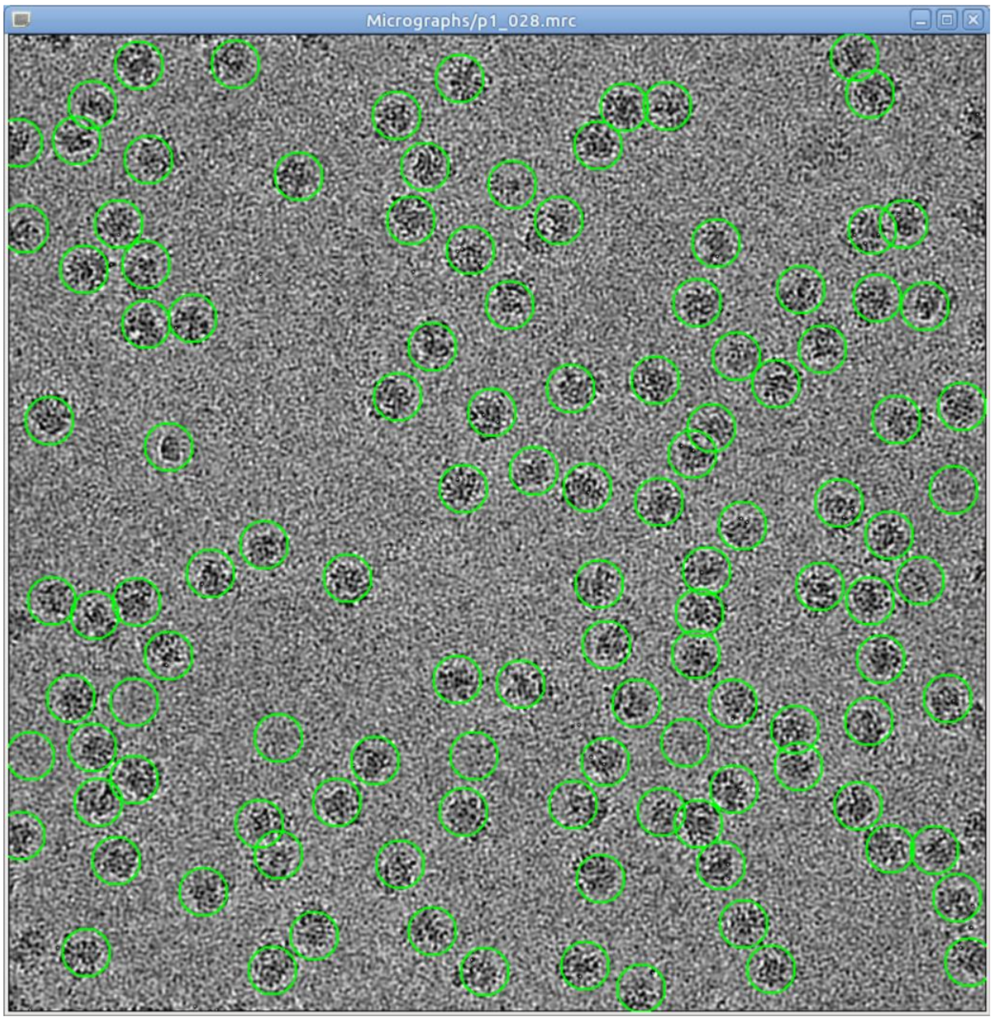


Figure 6: Display of the picked particles imported into `RELION`