Two Complete Architectures for Automatic Earthquake Detection and Location: *Part 2.* Running PAL & MESS

speaker: Yijian ZHOU[1]

Abhijit GHOSH[1], Lihua FANG[2], Han YUE[3], Shiyong ZHOU[3]

contact info: yijian.zhou@email.ucr.edu

UNIVERSITY OF CALIFORNIA UCRIVERSIDE

Outline

- Hardware & software
- Running PAL
- Running MESS

Outline

- **Hardware & software**
- Running PAL
- Running MESS

Hardware Requirements

- CPU
  - number of CPU threads > number of processes allocated
  - e.g. 4C8T CPU can allocate up to 8 parallel PAL processes
- GPU
  - Nvidia RTX series, suggest >4G GPU memory
  - e.g. RTX 3060 12G
- Memory & Hard drive
  - baseline >8G, suggest >16G memory
  - SSD is great, HDD also works

Hard Drive	
1~10 Tb	SATA

100M/s

Memory	
100 Gb	PCIe

15G/s

15G/s

Graphics Memory	
10 Gb	GDDR

100G/s

Software Requirements

- Anaconda
  - https://www.anaconda.com/products/individual
  - Numpy, Scipy etc. are included
- Obspy
  - https://github.com/obspy/obspy/wiki/Installation-via-Anaconda
  - processing seismic data
- Pytorch
  - https://pytorch.org/
  - deep learning architecture, GPU computation

“Install” PALM

- Setup software environment
  - suggest installing Anaconda & Obspy & Pytorch sequentially
- Download code from Github (latest release v2.3)
  - suggest saving at /home/user/software
- Install HypoInverse & HypoDD
  - suggest saving the binary file at /home/user/bin
- *Go through the example dataset*

Download Example Data

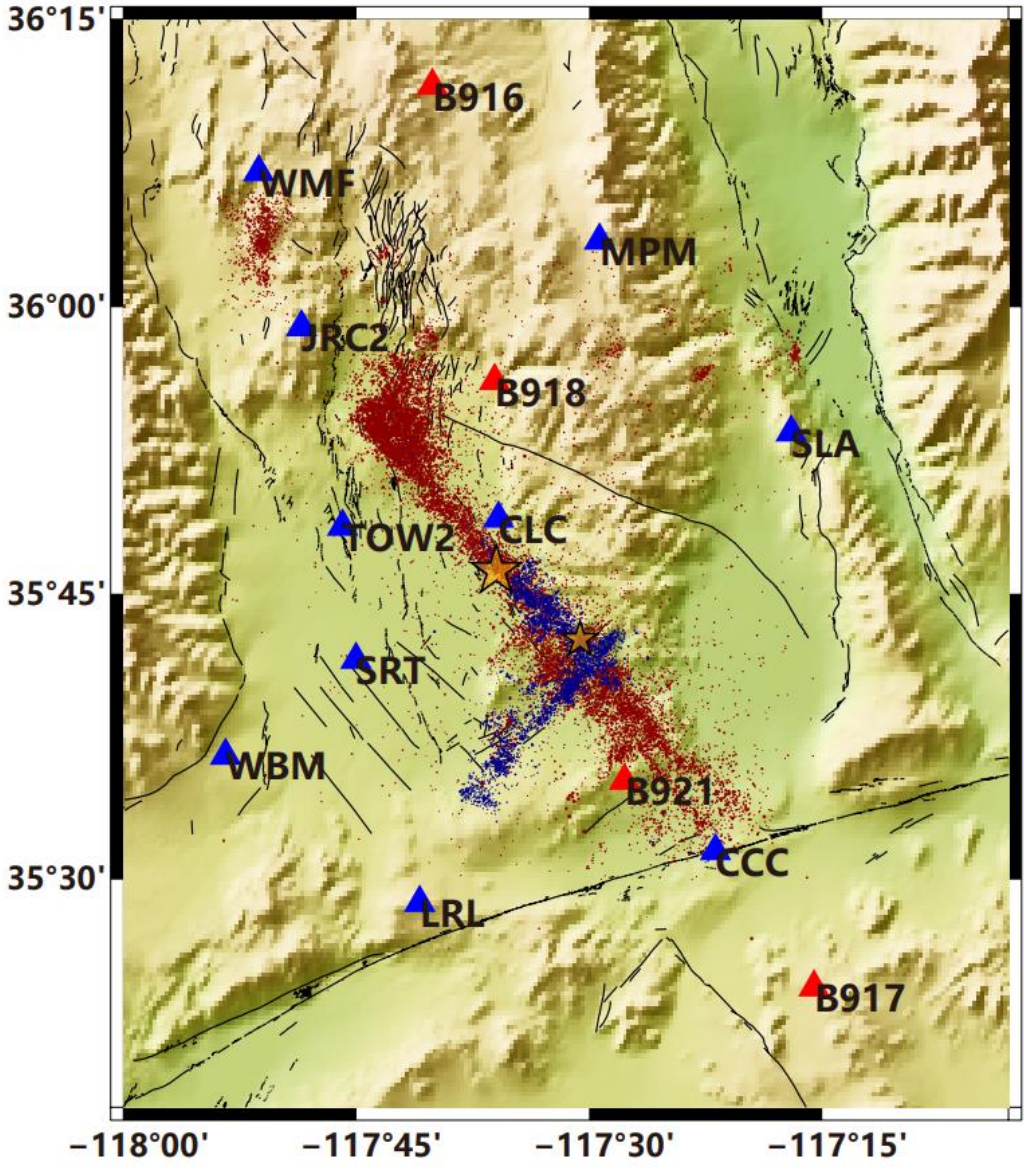
master ▾ PAL / example_pal_workdir / down_stp_data_eg.py / <> Jump to ▾

YijianZhou update example

1 contributor

52 lines (49 sloc) | 1.74 KB

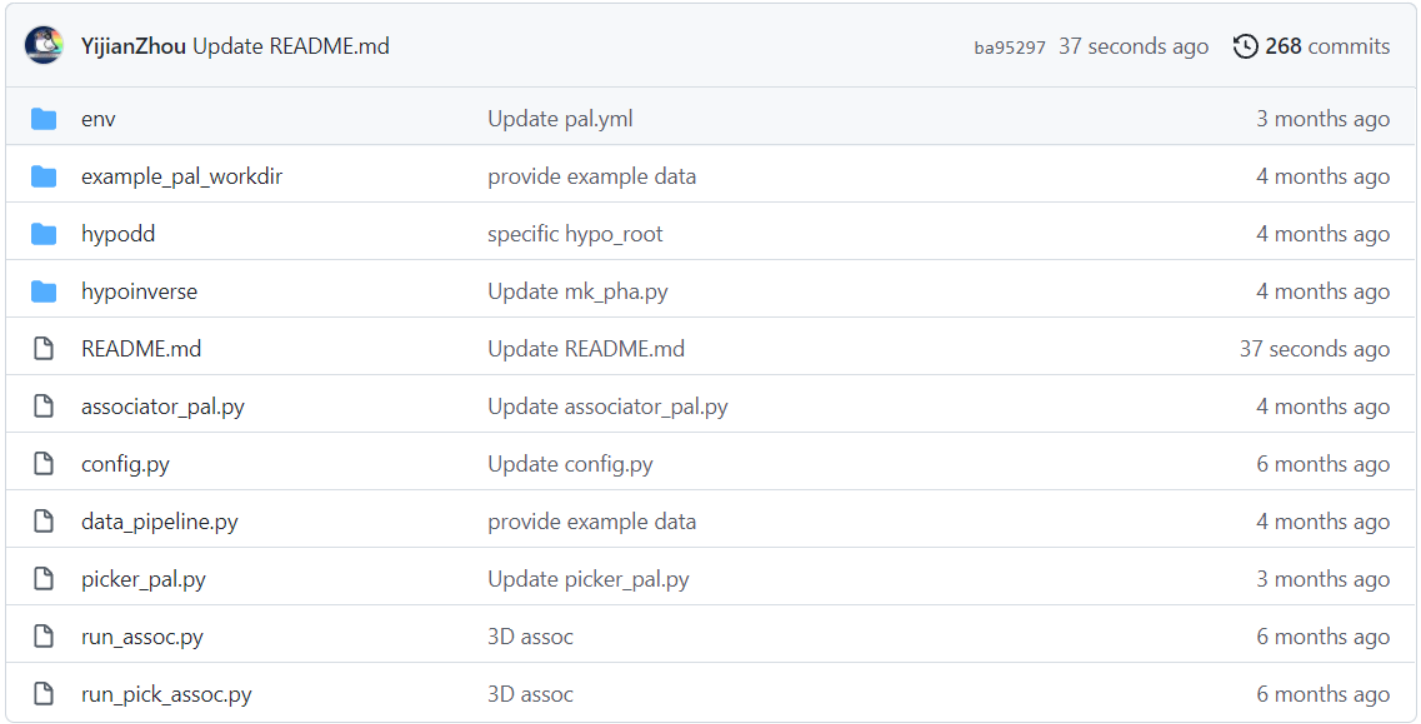
```
1   """ Download Example SCSN data by STP
2   STP can be downloaded from https://scedc.caltech.edu/data/stp/index.html
3   """
4   import os, shutil, glob
5   from obspy import UTCDateTime
6   import subprocess
7   import multiprocessing as mp
8
9   # i/o files
10  num_workers = 10
11  fsta = 'input/example_pal.sta'
12  time_range = '20190704-20190707'
13  out_root = '/data/Example_data'
14  if not os.path.exists(out_root): os.makedirs(out_root)
15  start_time, end_time = [UTCDateTime(date) for date in time_range.split('-')]
16  num_days = (end_time.date - start_time.date).days
```



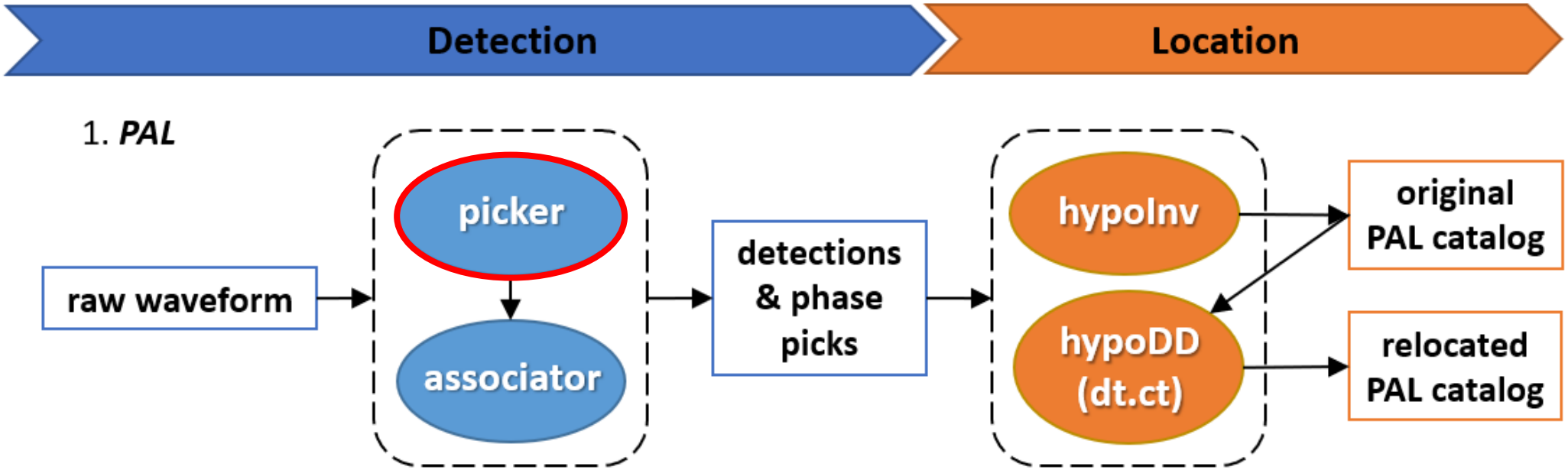
Outline

- Hardware & software
- **Running PAL**
- Running MESS

https://github.com/YijianZhou/PAL/releases/tag/v2.3



YijianZhou Update README.md	ba95297 37 seconds ago	**268** commits
env	Update pal.yml	3 months ago
example_pal_workdir	provide example data	4 months ago
hypodd	specific hypo_root	4 months ago
hypoinverse	Update mk_pha.py	4 months ago
README.md	Update README.md	37 seconds ago
associator_pal.py	Update associator_pal.py	4 months ago
config.py	Update config.py	6 months ago
data_pipeline.py	provide example data	4 months ago
picker_pal.py	Update picker_pal.py	3 months ago
run_assoc.py	3D assoc	6 months ago
run_pick_assoc.py	3D assoc	6 months ago

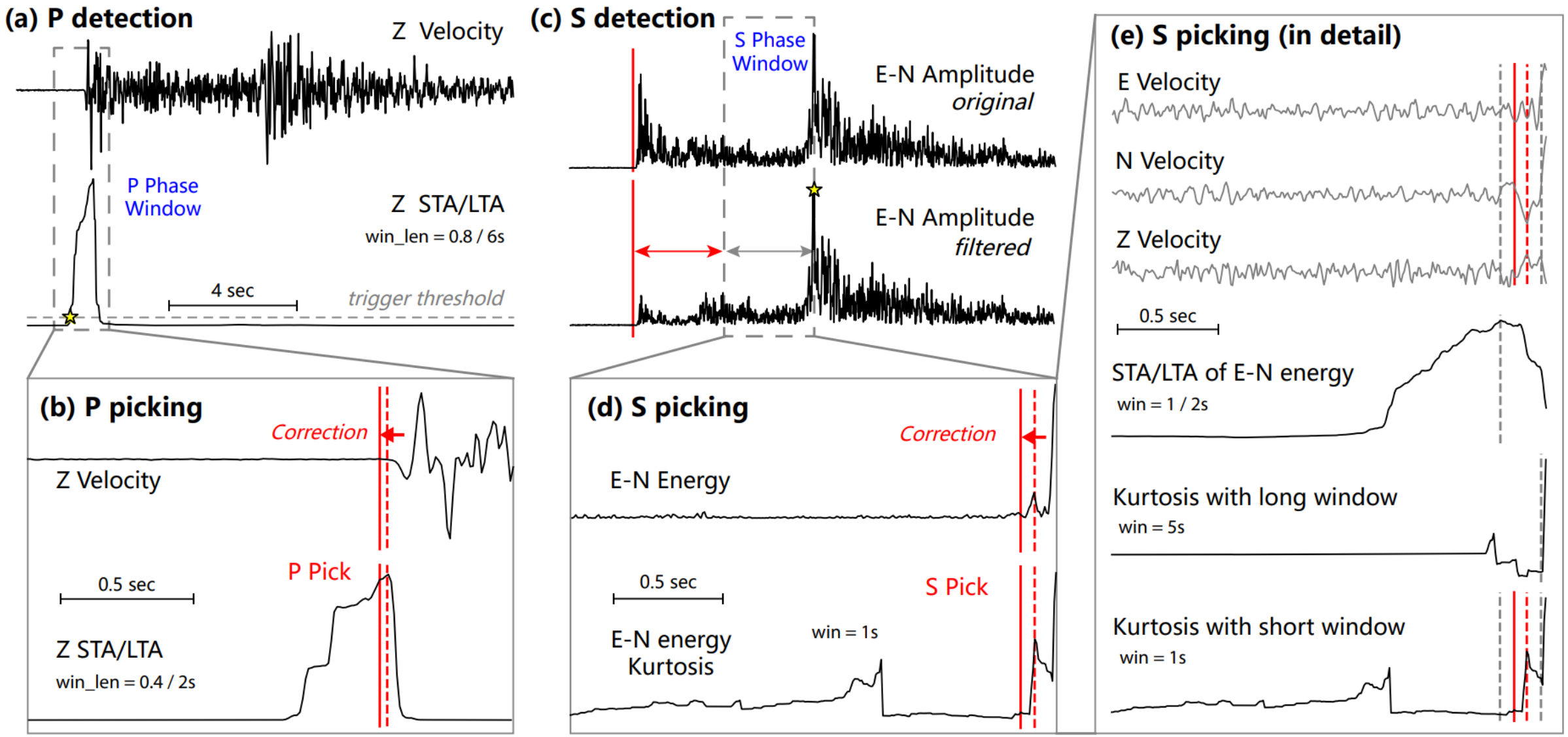


https://github.com/YijianZhou/PAL

Zhou et al., SRL 2021

PAL Config 1: Picking

```
# 1. picker params
self.win_sta    = [0.8,0.4,1.]  # win for STA: det, p, s
self.win_lta    = [6., 2., 2.]  # win for LTA: det, p, s
self.win_kurt   = [5.,1.]       # win for kurtosis: long & short
self.trig_thres = 12.           # threshold to trig picker (by energy)
self.p_win      = [.5,1.]       # search win for P
self.s_win      = 10.           # search win for S
self.pca_win    = 1.            # win_len for PCA filter
self.pca_range  = [0.,2.]       # time range to apply PCA filter
self.fd_thres   = 2.5           # min value of dominant frequency
self.amp_win    = [1.,4.]       # time win to get S amplitude
self.det_gap    = 5.            # time gap between detections
self.to_prep    = True          # whether to preprocess the raw data
self.freq_band  = [2,40]        # frequency band
```



Zhou et al., SRL 2021

PAL Config 1: Picking

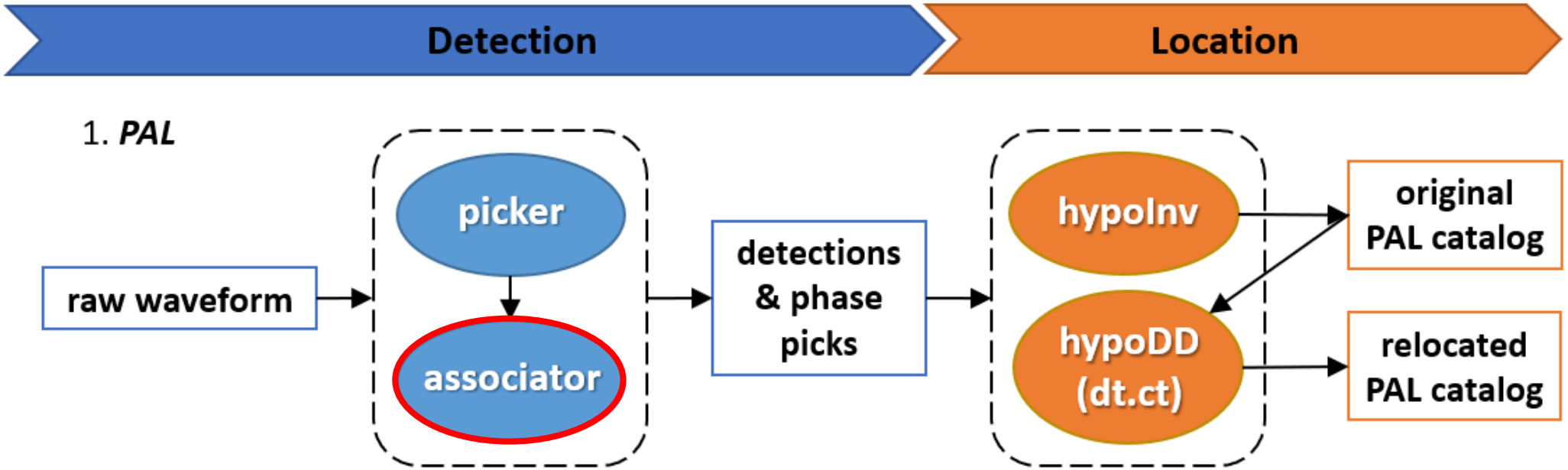
suitable for most cases

```
# 1. picker params
self.win_sta    = [0.8,0.4,1.]  # win for STA: det, p, s
self.win_lta    = [6.0,2.0,2.]  # win for LTA: det, p, s
self.win_kurt   = [5.,1.]       # win for kurtosis: long & short
self.trig_thres = 12.           # threshold to trig picker (by energy)
self.p_win      = [.5,1.]       # search win for P
self.s_win      = 10.           # search win for S
self.pca_win    = 1.            # win_len for PCA filter
self.pca_range  = [0.,2.]       # time range to apply PCA filter
self.fd_thres   = 2.5           # min value of dominant frequency
self.amp_win    = [1.,4.]       # time win to get S amplitude
self.det_gap    = 5.            # time gap between detections
self.to_prep    = True          # whether to preprocess the raw data
self.freq_band  = [2,40]        # frequency band
```

PAL Config 1: Picking

```
# 1. picker params
self.win_sta     = [0.8,0.4,1.]   # win for STA: det, p, s
self.win_lta     = [6., 2., 2.]   # win for LTA: det, p, s
self.win_kurt    = [5.,1.]        # win for kurtosis: long & short
self.trig_thres  = 12.            # threshold to trig picker (by energy)
self.p_win       = [.5,1.]        # search win for P
self.s_win       = 10.            # search win for S
self.pca_win     = 1.             # win_len for PCA filter
self.pca_range   = [0.,2.]        # time range to apply PCA filter
self.fd_thres    = 2.5            # min value of dominant frequency
self.amp_win     = [1.,4.]        # time win to get S amplitude
self.det_gap     = 5.             # time gap between detections
self.to_prep     = True           # whether to preprocess the raw data
self.freq_band   = [2,40]         # frequency band
```

Set picking params according to the *station distribution*
(average inter-distance etc.) & noise level

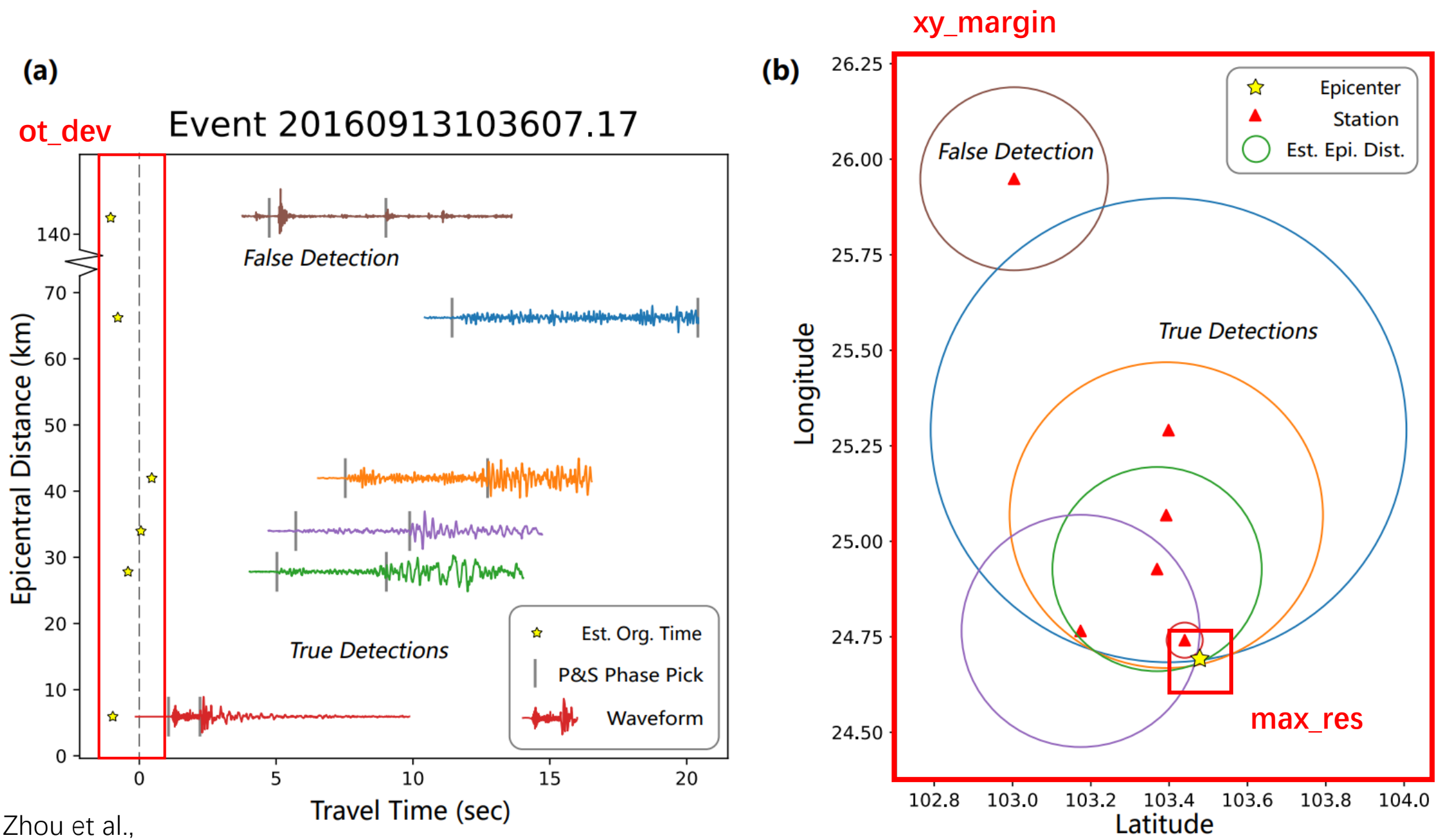


https://github.com/YijianZhou/PAL

Zhou et al., SRL 2021

PAL Config 2: Association

```
# 2. assoc params
self.min_sta     = 4          # min num of sta to assoc
self.ot_dev      = 2.         # max time deviation for ot assoc
self.max_res     = 1.5        # max P res for loc assoc
self.xy_margin   = 0.1        # xy (lateral) range inferred from sta loc
self.xy_grid     = 0.02       # xy (lateral) grid size (in degree)
self.z_grids     = [5]        # z (dep) grids (in km)
self.vp          = 5.9        # averaged P velocity
```



Zhou et al.,
SRL 2021

PAL Config 2: Association

```
# 2. assoc params
self.min_sta     = 4          # min num of sta to assoc
self.ot_dev      = 2.         # max time deviation for ot assoc
self.max_res     = 1.5        # max P res for loc assoc
self.xy_margin   = 0.1        # xy (lateral) range inferred from sta loc
self.xy_grid     = 0.02       # xy (lateral) grid size (in degree)
self.z_grids     = [5]        # z (dep) grids (in km)
self.vp          = 5.9        # averaged P velocity
```

Set assoc params according to the *station distribution* (average inter-distance etc.)

PAL Data_pipeline

1. Directory structure:
   data_root/yyyymmdd/net.sta.date.chn.sac
2. Require 3-channel, naming as ENZ/123 etc.
3. Modify *get_data_dict* if necessary

```
# get data path dict
def get_data_dict(date, data_dir):
    # get data paths
    data_dict = {}
    date_code = '{:0>4}{:0>2}{:0>2}'.format(date.year, date.month, date.day)
    st_paths = sorted(glob.glob(os.path.join(data_dir, date_code, '*')))
    for st_path in st_paths:
        fname = os.path.basename(st_path)
        net_sta = '.'.join(fname.split('.')[0:2])
        if net_sta in data_dict: data_dict[net_sta].append(st_path)
        else: data_dict[net_sta] = [st_path]
    # drop bad sta
    todel = [net_sta for net_sta in data_dict if len(data_dict[net_sta])!=3]
    for net_sta in todel: data_dict.pop(net_sta)
    return data_dict
```

PAL Data_pipeline

Format of station file:
net.sta, sta_lat, sta_lon, sta_ele, gain

```
# get station loc & gain dict
def get_sta_dict(sta_file):
    sta_dict = {}
    dtype = [('sta_lat','O'),('sta_lon','O'),('sta_ele','O'),('gain','O')]
    f=open(sta_file); lines = f.readlines(); f.close()
    for line in lines:
        codes = line.split(',')
        net_sta = codes[0]
        lat, lon, ele, gain = [float(code) for code in codes[1:5]]
        sta_dict[net_sta] = np.array((lat,lon,ele,gain), dtype=dtype)
    return sta_dict
```

Running PAL

- Pick + Assoc: ***parallel_pick_assoc.py***
- Assoc only: ***parallel_assoc.py***

```
pal_dir = '/home/zhouyj/software/PAL'
shutil.copyfile('config_eg.py', os.path.join(pal_dir, 'config.py'))
data_dir = '/data/Example_data'
time_range = '20190704-20190707'
sta_file = 'input/example_pal.sta'
num_workers = 3
out_root = 'output/eg'
out_pick_dir = 'output/eg/picks'
```

1. Parallel by time, the minimum unit is 1 day
2. Set date for *time_range*, but they are used as time (*end date + 1*)
3. Run *python parallel_pick_assoc.py*
4. In the output dir: *cat phase_* > pal.pha*

PAL Output

- PAL output picks
  - fpath: out_root/picks/yyyy-mm-dd.pick
  - format: net.sta, ot, tp, ts, s_amp, p_snr, freq_dom

- PAL output phase file
  - fpath: out_root/phase_yyyymmdd-yyyymmdd.dat
  - event line: ot, lat, lon, dep, mag
  - phase line: net.sta, tp, ts, s_amp, p_snr

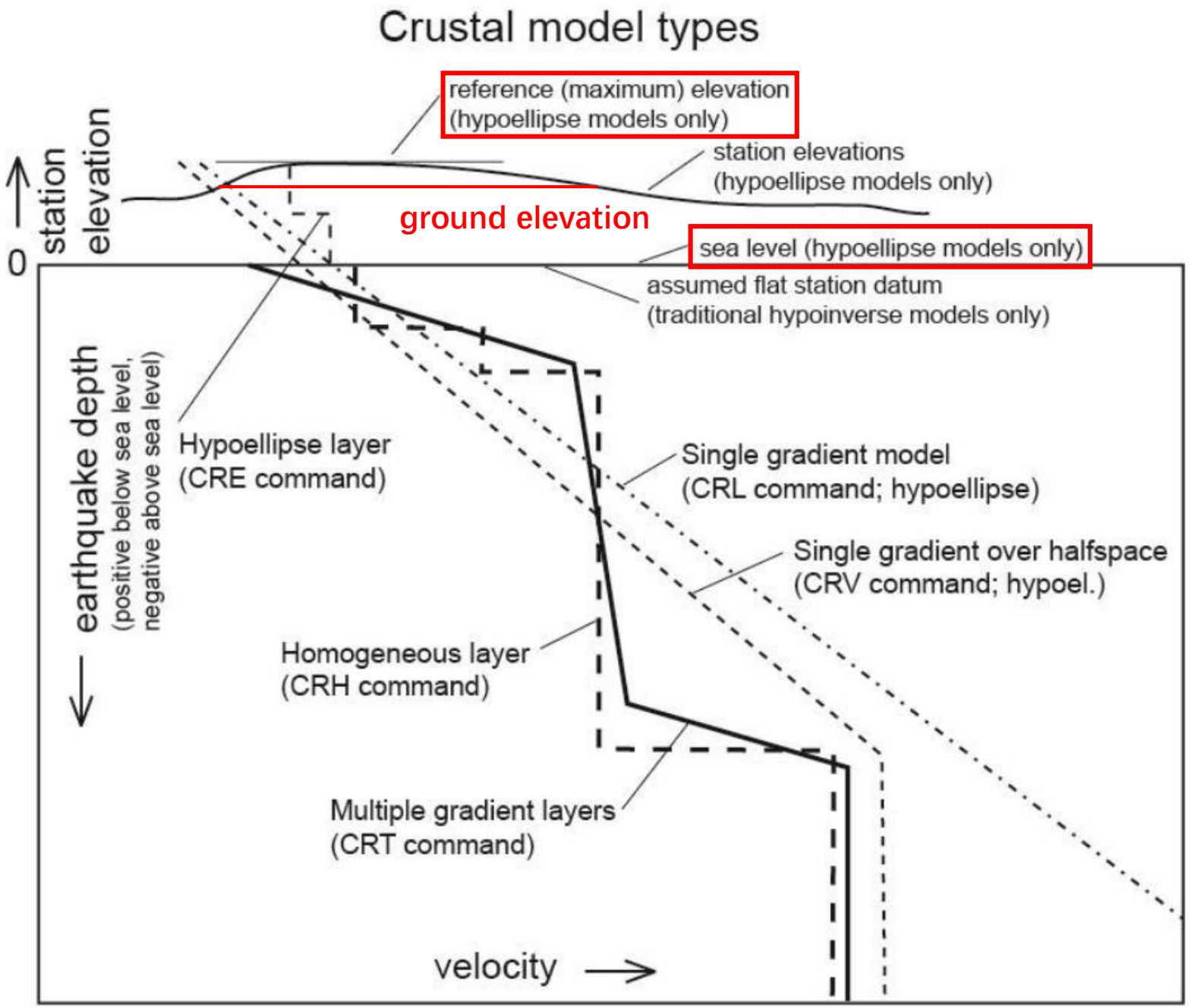
Pick-based Location

- HypoInverse
  - Software: https://www.usgs.gov/software/hypoinverse-earthquake-location
  - Document: https://doi.org/10.3133/ofr02171

- HypoDD (dt.ct)
  - Software: https://www.ldeo.columbia.edu/~felixw/hypoDD.html
  - Document: https://www.ldeo.columbia.edu/~felixw/papers/Waldhauser_OFR2001.pdf

Run HypoInverse

- Station file
  - same as PAL, just copy into input/
- Velocity model
  - in CRE format, which supports station elevation
  - set *ref_ele* & *grd_ele*, make necessary correction
- Location parameters
  - weighting by distance
  - weighting by residual


Crustal model types
reference (maximum) elevation
(hypoellipse models only)
station elevations
(hypoellipse models only)
ground elevation
sea level (hypoellipse models only)
assumed flat station datum
(traditional hypoinverse models only)
0
station elevation
earthquake depth
(positive below sea level,
negative above sea level)
Hypoellipse layer
(CRE command)
Single gradient model
(CRL command; hypoellipse)
Single gradient over halfspace
(CRV command; hypoel.)
Homogeneous layer
(CRH command)
Multiple gradient layers
(CRT command)
velocity

Klein, 2014

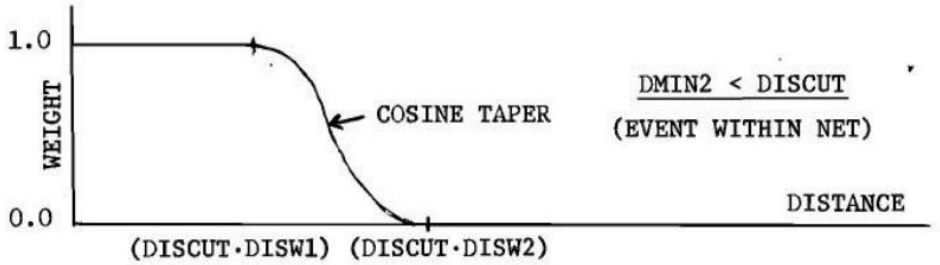
```
self.ctlg_code = 'eg_pal_hyp'
# 1. mk_sta: format station file
self.fsta = 'input/example_pal.sta'
self.lat_code = 'N'
self.lon_code = 'W'
# 2. mk_pha: format phase file
self.fpha = 'input/eg_pal.pha’
# 3. sum2csv: format output files
self.ref_ele = 2.5 # ref ele for CRE mod (max sta ele)
self.grd_ele = 1.5 # typical station elevation
# 4. run_hyp
self.ztr_rng = np.arange(0,20,1)
self.p_wht = 0 # weight code
self.s_wht = 1
self.rms_wht = '4 0.3 1 3'
self.dist_init = '1 50 1 2'
self.dist_wht = '4 20 1 3'
self.wht_code = '1 0.6 0.3 0.2'
self.fhyp_temp = 'temp_hyp/temp_vp-pos.hyp'
self.pmod = 'input/example_p.cre'
self.smod = [None, 'input/example_s.cre'][0]
self.pos = 1.73 # provide smod or pos
```

Write CRE Velocity Model

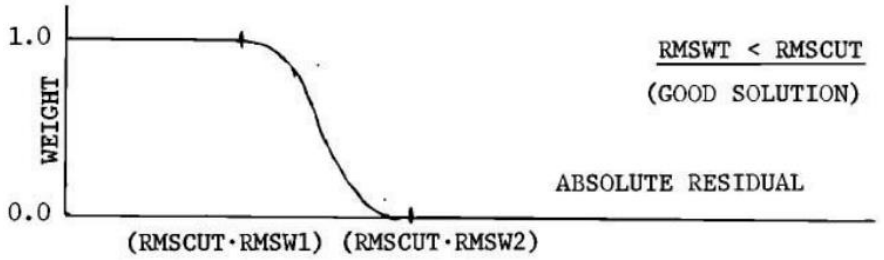
HK Model with V_P/V_S Ratio of 1.73		
Depth to Top of Layer (km)	CRE Interfaces	*P*-Velocity (km/sec)
0.0	0.0	5.5
5.5	6.5	6.3
16.0	17.0	6.7
32.0	33.0	7.8

Weighting Scheme

- Distance weighting
  - <min_dist: full weight
  - >max_dist: zero weight
  - min_dist~max_dist: cos taper



- Residual weighting
  - <min_res: full weight
  - >max_res: zero weight
  - min_res~max_res: cos taper



Klein, 2014

```
self.ctlg_code = 'eg_pal_hyp'
# 1. mk_sta: format station file
self.fsta = 'input/example_pal.sta'
self.lat_code = 'N'
self.lon_code = 'W'
# 2. mk_pha: format phase file
self.fpha = 'input/eg_pal.pha’
# 3. sum2csv: format output files
self.ref_ele = 2.5 # ref ele for CRE mod (max sta ele)
self.grd_ele = 1.5 # typical station elevation
# 4. run_hyp
self.ztr_rng = np.arange(0,20,1)
self.p_wht = 0 # weight code
self.s_wht = 1
self.rms_wht = '4 0.3 1 3'
self.dist_init = '1 50 1 2'
self.dist_wht = '4 20 1 3'
self.wht_code = '1 0.6 0.3 0.2'
self.fhyp_temp = 'temp_hyp/temp_vp-pos.hyp'
self.pmod = 'input/example_p.cre'
self.smod = [None, 'input/example_s.cre'][0]
self.pos = 1.73 # provide smod or pos
```

HypoInverse Output

- Catalog: *eg_pal_hyp.ctlg*
  - format: ot, lat, lon, dep, mag
- Phase: *eg_pal_hyp.pha*
  - event line: ot, lat, lon, dep, mag
  - phase line: net.sta, tp, ts, s_amp, p_snr
- Phase with evid: *eg_pal_hyp_full.pha*
  - event line: ot, lat, lon, dep, mag, evid
  - phase line: net.sta, tp, ts, s_amp, p_snr
- Quality control files: *.sum, _good.csv, _bad.csv*

Run HypoDD

- Station file
  - same as PAL, just *cp* into *input/*
- Phase file
  - _full.pha, output of HypoInverse
  - event line: ot, lat, lon, dep, mag, *evid*
  - phase line: net.sta, tp, ts
- Location parameters
  - ph2dt
  - hypoDD

```
self.hypo_root = '/home/zhouyj/bin'
self.ctlg_code = 'eg_pal_ct'
self.fsta = 'input/example_pal.sta'
self.fpha = 'input/eg_pal_hyp_full.pha'
self.dep_corr = 5 # avoid air quake
self.ot_range = '20190704-20190707'
self.lat_range = [35.45,36.05]
self.lon_range = [-117.8,-117.25]
self.num_grids = [1,1] # x,y (lon, lat)
self.xy_pad = [0.06,0.05] # degree
self.num_workers = 5
```

hypoDD -- A Program to Compute Double-Difference Hypocenter Locations

(*hypoDD* version 1.0 - 03/2001)

by

Felix Waldhauser

U.S. Geol. Survey
345 Middlefield Rd, MS977
Menlo Park, CA 94025
felix@andreas.wr.usgs.gov

Please read this document!

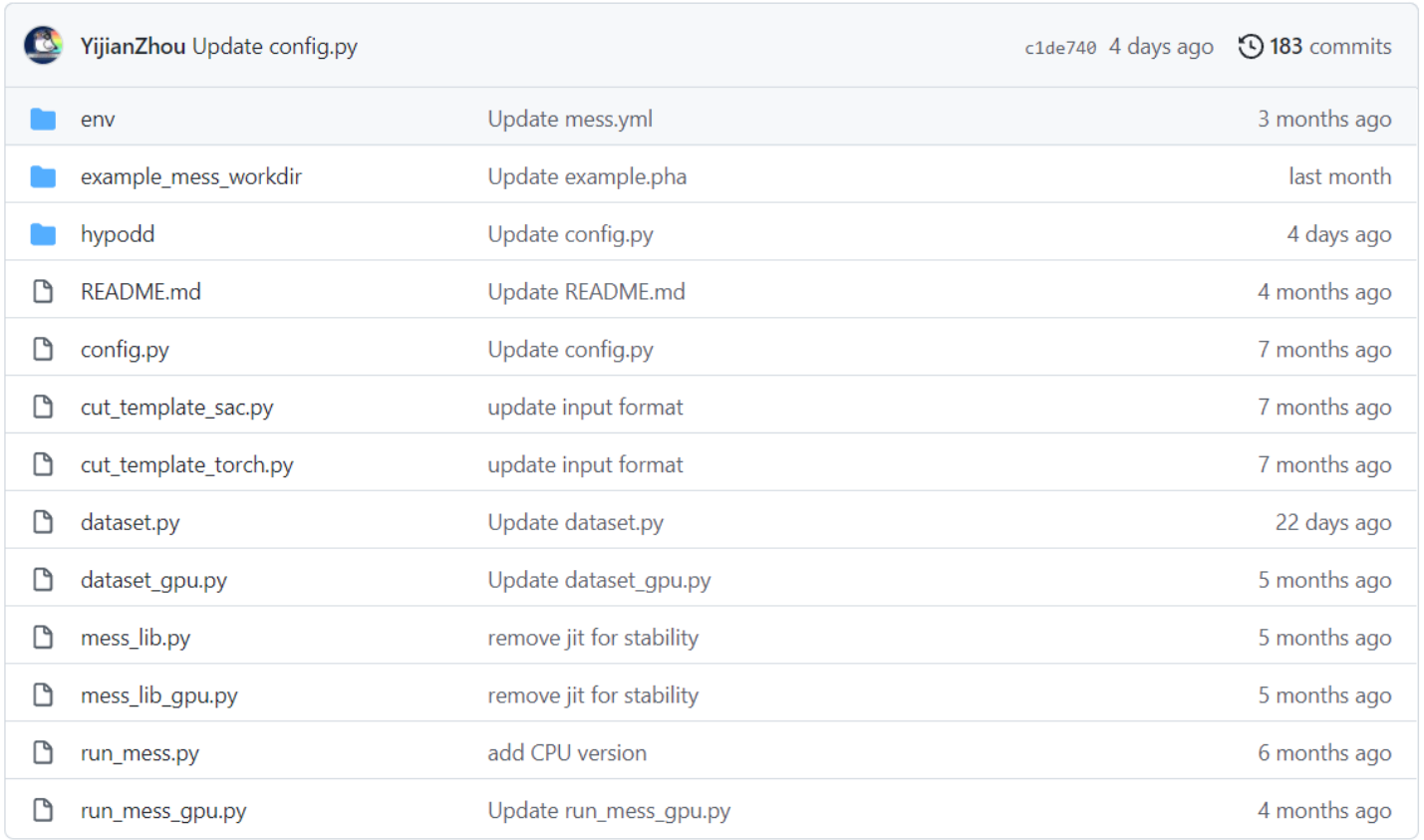
HypoDD Output

- Catalog: *eg_pal_ct.ctlg*
  - format: ot, lat, lon, dep, mag
- Phase: *eg_pal_ct.pha*
  - event line: ot, lat, lon, dep, mag
  - phase line: net.sta, tp, ts, s_amp, p_snr
- Phase with evid: *eg_pal_ct_full.pha*
  - event line: ot, lat, lon, dep, mag, evid
  - phase line: net.sta, tp, ts, s_amp, p_snr
- Quality control files: *.ph2dt & .hypoDD (screen output)*

Outline

- Hardware & software
- Running PAL
- **Running MESS**

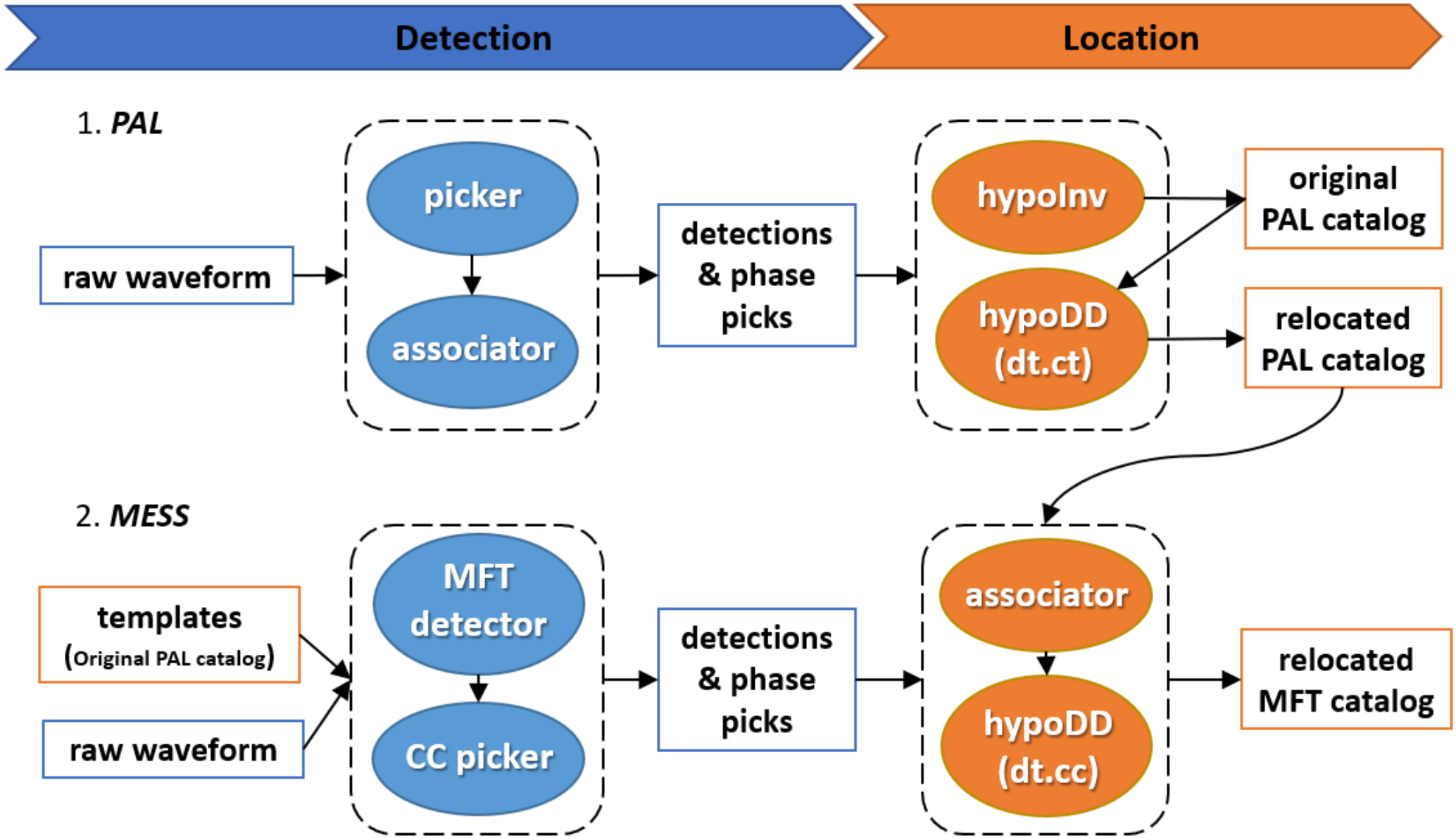
https://github.com/YijianZhou/MESS/releases/tag/v2.3



YijianZhou Update config.py — c1de740 4 days ago — 183 commits

env	Update mess.yml	3 months ago
example_mess_workdir	Update example.pha	last month
hypodd	Update config.py	4 days ago
README.md	Update README.md	4 months ago
config.py	Update config.py	7 months ago
cut_template_sac.py	update input format	7 months ago
cut_template_torch.py	update input format	7 months ago
dataset.py	Update dataset.py	22 days ago
dataset_gpu.py	Update dataset_gpu.py	5 months ago
mess_lib.py	remove jit for stability	5 months ago
mess_lib_gpu.py	remove jit for stability	5 months ago
run_mess.py	add CPU version	6 months ago
run_mess_gpu.py	Update run_mess_gpu.py	4 months ago

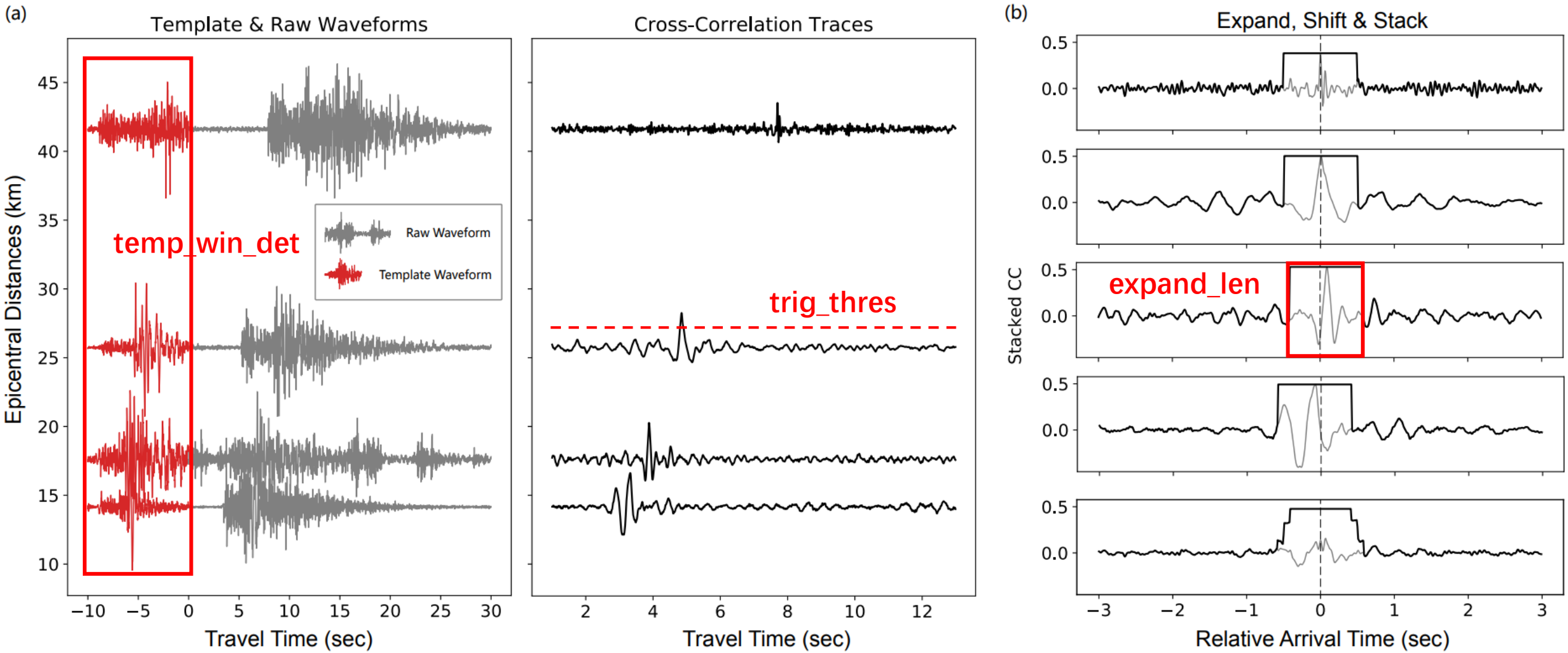
UNIVERSITY OF CALIFORNIA UCRIVERSIDE



https://github.com/YijianZhou/MESS

Zhou et al., SRL 2021

```
# MFT params
self.win_len = [10,20]          # cut template length
self.min_sta = 4                # min sta num for template event
self.max_sta = 15               # max sta num for template event
self.temp_win_det = [1.,9.]     # temp win for detection, pre & post P
self.temp_win_p = [0.5,1.5]     # temp win for p pick, pre & post P
self.temp_win_s = [0.5,2.5]     # temp win for s pick, pre & post S
self.trig_thres = 0.3           # cc thres for det & peak expansion
self.expand_len = 1.            # win len for cc peak expansion
self.det_gap = 5.               # gap sec for detection
self.pick_win_p = [1.0,1.0]     # search win for P pick
self.pick_win_s = [1.6,1.6]     # search win for S pick
self.chn_p = [2]                # chn for P pick
self.chn_s = [0,1]              # chn for S pick
self.amp_win = [1,4]            # win for amp measurement
# data process
self.samp_rate = 50
self.freq_band = [2.,40.]
self.num_workers = 10
self.get_data_dict = dp.get_data_dict
self.get_sta_dict = dp.get_sta_dict
```



Zhou et al., SRL 2021

```
# MFT params
self.win_len = [10,20]          # cut template length
self.min_sta = 4                # min sta num for template event
self.max_sta = 15               # max sta num for template event
self.temp_win_det = [1.,9.]     # temp win for detection, pre & post P
self.temp_win_p = [0.5,1.5]     # temp win for p pick, pre & post P
self.temp_win_s = [0.5,2.5]     # temp win for s pick, pre & post S
self.trig_thres = 0.3           # cc thres for det & peak expansion
self.expand_len = 1.            # win len for cc peak expansion
self.det_gap = 5.               # gap sec for detection
self.pick_win_p = [1.0,1.0]     # search win for P pick
self.pick_win_s = [1.6,1.6]     # search win for S pick
self.chn_p = [2]                # chn for P pick
self.chn_s = [0,1]              # chn for S pick
self.amp_win = [1,4]            # win for amp measurement
# data process
self.samp_rate = 50
self.freq_band = [2.,40.]
self.num_workers = 10
self.get_data_dict = dp.get_data_dict
self.get_sta_dict = dp.get_sta_dict
```

Set MESS params based on the *station distribution* (average inter-distance etc.)

Running MESS

- Make template phase
  - PAL det phase: *eg_pal.pha* → evid & event name
  - HypoInverse phase: *eg_pal_hyp.pha* → location
- Cut templates
  - *cut_templates_obspy.py:* if many events in one day
  - *cut_templates_sac.py:* if few events in one day
- Run MESS
  - use GPU version to enjoy ×40 acceleration
  - use CPU version if you have >100 cores cluster

1. Make Template Phase: *pha2temp.py*

- Inputs
  - Detection phase: *eg_pal.pha*, providing evid & event name
  - Located phase: *eg_pal_hyp_full.pha*
- Outputs
  - Template phase: *evid_name, ot, lat, lon, dep, mag*

```
# i/o paths
fpha_det = 'input/eg_pal.pha'
fpha_loc = 'input/eg_pal_hyp_full.pha'
fout = open('input/eg_pal.temp','w')
# selection criteria
ot_range = '20190704-20190707'
ot_range = [UTCDateTime(code) for code in ot_range.split('-')]
lat_range = [35.5,36.]
lon_range = [-117.8,-117.3]
```

2. Cut Templates

- Cut with Obspy slice: *cut_templates_obspy.py*
  - read 1-sta 1-day's data & preprocess
  - slice all phases on that station & that day
- Cut with SAC cut: *cut_templates_sac.py*
  - for all events,
  - read ~30s event window & preprocess

~100,000 events / ~1,000,000 phases, finish with 20min

```
# i/o paths
mess_dir = '/home/zhouyj/software/MESS'
data_dir = '/data/Example_data'
out_root = 'output/Example_templates'
temp_pha = 'input/eg_pal.temp'
```

3. Running MESS

- Can start multiple processes based on the GPU memory
  - n_sta * 3 chn * 4 bit * 86400 sec * samp_rate
  - → if 50Hz, 100M * n_sta
- Can start multiple processes based on the CPU threads

```
# i/o paths
gpu_idx = '0'
mess_dir = '/home/zhouyj/software/MESS'
data_dir = '/data/Example_data'
time_range = '20190704-20190707'
sta_file = 'input/example_pal.sta'
temp_root = 'output/Example_templates'
temp_pha = 'input/eg_pal.temp'
```

```
template 0_20190704161342.98
15 detections, 10 stations, 0.8s
det_ot 2019-07-04T15:35:29.700000Z, det_cc 0.30
CI.TOW2 | dt_p -0.02s, dt_s -0.02s | cc_p 0.716, cc_s 0.744
CI.SRT  | dt_p -0.02s, dt_s 0.00 s | cc_p 0.395, cc_s 0.410
CI.CCC  | dt_p -0.02s, dt_s -0.02s | cc_p 0.378, cc_s 0.567
CI.MPM  | dt_p 0.00 s, dt_s 0.00 s | cc_p 0.849, cc_s 0.429
CI.JRC2 | dt_p 0.00 s, dt_s -0.02s | cc_p 0.334, cc_s 0.669
CI.CLC  | dt_p -0.02s, dt_s -0.02s | cc_p 0.558, cc_s 0.599
CI.SLA  | dt_p -0.56s, dt_s 0.00 s | cc_p 0.370, cc_s 0.382
CI.LRL  | dt_p 0.40 s, dt_s -0.02s | cc_p 0.265, cc_s 0.280
CI.WBM  | dt_p -0.22s, dt_s -0.90s | cc_p 0.255, cc_s 0.208
```

HypoDD Relocation

- Relocate templates (*pal_ct_full.pha*)
- Relocate MESS detections: *python run_hypoDD.py*
  - associate detections from different templates → *dt.cc*
  - run hypoDD (relocate with *dt.cc*)

```
# 2. mk_dt
self.temp_pha = 'input/eg_pal_ct_full.pha'    # reloc template phase file
self.det_pha = 'input/eg_mess.pha'    # mess output phase file
self.ot_dev = 2.    # ot diff for det assoc
self.cc_thres = [0.3,0.4]    # CC thres for det & pick
self.dt_thres = [0.6,1.]    # max dt_p & dt_s
self.nbr_thres = [2,30]    # min & max num of neighbor event
self.min_sta = 4
# 3. reloc2csv
self.lat_range = [35.4,36.1]
self.lon_range = [-117.85,-117.25]
self.xy_pad = [0.046,0.037]    # degree
self.num_grids = [1,1]    # x,y (lon, lat)
```

MESS Output

- “Phase” output: *mess.pha*
  - fpath: out_root/phase_yyyymmdd-yyyymddd.dat
  - event line: evid_name, ot, lat, lon, dep, cc
  - phase line: net.sta, tp, ts, dt_p, dt_s, s_amp, cc_p, cc_s

- HypoDD output catalog: *mess_cc.ctlg*
  - format: ot, lat, lon, dep, mag

Summary

- Prepare data
  - consistent directory structure → whether modify *data_pipeline.py*
  - plot station location → reselection of station & tune PALM parameters

- Run PALM
  - rewrite file & directories
  - inspect detection and location results

Remaining Topics

- Complex network configuration
- Tuning detection parameters
- Tuning location parameters
- Combination of different modules
- Special datasets (e.g. dense array, OBS, DAS etc.)
- … …

References

- **Zhou, Y.**, H. Yue, L. Fang, S. Zhou, L. Zhao, & A. Ghosh (2021). An Earthquake Detection and Location Architecture for Continuous Seismograms: Phase Picking, Association, Location, and Matched Filter (PALM). Seismological Research Letters. doi: https://doi.org/10.1785/0220210111
- **Zhou, Y.**, A. Ghosh, L. Fang, H. Yue, & S. Zhou (2021, *in preparation*). Early Aftershock Catalog of the 2019 Ridgecrest, California Earthquake: Application of an AI-based Detection and Location Architecture.
- **Zhou, Y.**, H. Yue, Q. Kong, & S. Zhou (2019). Hybrid Event Detection and Phase-Picking Algorithm Using Convolutional and Recurrent Neural Networks. Seismological Research Letters; 90 (3): 1079–1087. doi: https://doi.org/10.1785/0220180319
- **Zhou, Y.**, A. Ghosh, L. Fang, H. Yue, S. Zhou, & Y. Su (2021). A High-Resolution Seismic Catalog for the 2021 MS6.4/Mw6.1 YangBi Earthquake Sequence, Yunnan, China: Application of AI picker and Matched Filter. *Earthquake Science*; doi: 10.29382/eqs-2021-0031

Contact Us!

Include but not limited to:
Report bugs
Assistance in usage
Cooperation
Technical discussion
… …

周一剑 Yijian ZHOU
Email: yijian.zhou@email.ucr.edu
WeChat: zhouyj_observer
Github: https://github.com/YijianZhou

PALM用户交流服务群

Valid until 11/4 and will update upon joining group