targeted spatio-temporal sampling using raytraverse
Schedule / Goals

- Introduction (20 min):
  + what is raytraverse?
  + How does it work?
- Exercise #1 (30 min): getting started
- Setting up a Simulation (20 min)
  + input data
  + CBDM methods
- Exercise #2 (30 min): hourly metrics from a point
- ----- Break ----- 
- Simulation Parameters (10 min)
- basic scripting (20 min)
- Exercise #3 (30 min): sensor area sampling
introduction
What is raytraverse?

**Method**
- Scene model: sky, material, etc.
- Observer model: comfort metric, tone-mapping, etc.
- Renderer: solve for a view-ray, i.e., Radiance
- Evaluation: apply observer model to results
- Interpretation: visualization, summary metrics, etc.

**Python Library**
- `from raytraverse.scene import Scene`
- `from raytraverse.renderer import Rtrace, Rcontrib`
- `from raytraverse.sampler import SamplerP1, SamplerArea`
- `from raytraverse.mapper import ViewMapper, SkyMapper, PlanMapper`
- `from raytraverse.lightpoint import LightPointKD`
- `from raytraverse.lightfield import LightPlaneKD, LightResult`
- `from raytraverse.integrator import Integrator`
- `from raytraverse.evaluate import MetricSet`

**Workflow**
- Application, tool, or instructions for running the process

**Parameters**
- Input: scene, geometry, material, and light source specification
- Sampler: generates discrete samples as a representation of scope
- Renderer: solve for a view-ray, i.e., Radiance
- Evaluation: apply observer model to results
- Interpretation: visualization, summary metrics, etc.

**Output**
- Raytraverse methods shown in orange

**Usage:**
`raytraverse [OPTIONS] COMMAND1 [ARGS]... [COMMAND2]`

**Commands:**
- `area`: define sampling area
- `directs ayrun`: define scene files for renderer and output
- `evaluate`: evaluate metrics
- `images`: render images
- `pull`: define sky conditions for evaluation
- `skyengine`: initialize engine for skyrun
- `skyr un`: run scene under sky for a set of points
- `sourceengine`: initialize engine for sunrun
- `sourcerun`: run scene for a single source
- `sunengine`: initialize engine for sunrun
- `sunrun`: run scene for a set of suns
- `suns`: define solar sampling space
Why is sampling so important?

If one pixel is the volume of a studio apartment....

Direction $\times 10^6$

Position $\times 10^2$

Time / Source $\times 10^3$

Spectrum $\times 10^1$

Pluto - Ganymede

Mars - Neptune

Saturn - Sun

Jupiter - Sun+
Organizing the Sampling Dimensions to efficiently encode useful variance

Tregenza (Reinhart and Klems are similar)

A simple 2D grid

Shirley-Chiu Low distortion map between disk and square
Shirley-Chiu - Useful for Sampling and DWT

Dual Shirley-Chiu
Preserves fractional area
Bi-continuous
Low distortion

Dual Equiangular Fisheyes
Preserves fractional area
Bi-continuous
Low distortion

https://en.wikipedia.org/wiki/Equirectangular_projection
Wavelet Guided Sampling

Compression

32x32  3x32x32  3x64x64  3x128x128  3x256x256  3x512x512  1024x1024

Reconstruction

Sampling

32x32  64x64  128x128  256x256  512x512  1024x1024  1024x1024
Sampling Directions from a Point
Sampling Sun Positions

samples average luminance GCR detail

qualitative color scale

low

high
What does this get us?
If one ray is the volume of a studio apartment....
Sampling an Area and Sun Positions

3 levels of sun sampling for full zone

point samples for level 0 suns
Sampling an Area and Sun Positions

3 levels of sun sampling in high variance region
Sampling an Area and Sun Positions

3 levels of sun sampling in low variance region
Exercise #0: prerequisites

Windows
- Install Docker from: https://www.docker.com/products/docker-desktop/
  + click on “Windows” and then follow the installation instructions.
- Open the newly installed Docker Desktop application
  + you do not need to sign in or create an account
- download and unzip <<rwt>>
- open a command prompt and navigate to rwt/
- docker build . --tag rwt:latest
  + copy text from Dockerfile

Macos/Linux
- (as an alternative, on a mac, you can follow the windows instructions)
- Ensure python3.7 - python 3.10 is installed on your machine
  + check in a terminal with: python3 --version
  + for mac, I suggest using homebrew: (find tutorial link)
  + For ubuntu, make sure python has pip (find commands for this)
- install radiance: (github url)
- set path/raypath/manpath in .bash_profile
- open a terminal/command prompt and navigate to rwt/
- python3 -m venv rwtenv
- source rwtenv/bin/activate
- pip install -r Requirements.txt

see README.rst in rwt
exercise #1
Help

Usage: raytraverse [OPTIONS] COMMAND [ARGS]... [COMMAND [ARGS]...]

The raytraverse command is a command line interface to the raytraverse Python package for creating and evaluating climate based daylight models. Both commands of raytraverse can be invoked but should be invoked in the order given. The easiest way to manage options is to use a configuration file, to make a template:

```
raytraverse --template > ray.cfg
```

After adjusting the settings, each command can be invoked in turn and any dependencies will be loaded with the correct options. A complete run and evaluation can then be called by:

```
raytraverse --run.cfg
```

Any of the reporter modules can be evaluated as all required processor commands will be invoked automatically as needed.

Options:

- `--config`, `-c`
  - path of config file to load

- `-n INTERFACE`
  - set the environment variable RAYTRAVEL_PROC_DIR to 0 to allow parallel processes will use cpucounts

- `-o OUT_DIRECTORY`

Flags (default: false):

- `--template`, `-t`
  - write default options to std out as config (default: no-template)

Help:

- `-h`, `--help`
  - print this help message and exit. (default: False)

- `--version`
  - show the version and exit. (default: False)

Commands:

- `area`
- `directivity`
- `evaluate`
- `matrix`
- `render`
- `scene`
- `sensors`
- `sersic`
- `sersicfringe`
- `sersicline`
- `sersicpoint`
- `sersicprofile`
- `sersicfringe`
- `sersicline`
- `sersicpoint`
- `sersicprofile`
- `sersicfringe`
- `sersicline`
- `sersicpoint`
- `sersicprofile`
- `sersicfringe`
- `sersicline`
- `sersicpoint`
- `sersicprofile`
- `sersicfringe`
- `sersicline`
- `sersicpoint`
- `sersicprofile`
- `sersicfringe`
- `sersicline`
- `sersicpoint`
- `sersicprofile`
- `sersicfringe`
- `sersicline`
- `sersicpoint`
- `sersicprofile`
Exercise #1.a: Up and Running

- open a terminal/command prompt and navigate to rwt/
- for windows users (copy from ./Dockerfile):
  + make sure docker is running
  + docker run -it --name rayt --mount type=bind,source="$(pwd)",target=/working rwt /bin/bash
- for mac/linux:
  + source rwt/bin/activate
- cd ex1
- raytraverse --version

see README.rst in rwt
Exercise #1.b: Configuration File

raytraverse scene --opts

Options:
- debug............. False
- log............... True
- opts.............. False
- out............... None
- overwrite........ False
- reload............ True
- scene............... None

rayt.cfg:

[raytraverse_scene]
out = office
scene = office_glz.rad

[raytraverse_area]
name = vp1
static_points = 0.6,1,1.2

[raytraverse_sourceengine]
source = may3pm
rayargs = -ab 4
srcfile = MAY03-1530.rad

[raytraverse_images]
sensors = 0.6,1,1.2,1,0,0
basename = default
simtype = may3pm

Options:
- debug............. False
- log............... True
- opts.............. False
- out............... office
- overwrite........ False
- reload............ True
- scene............... office_glz.rad
Exercise #1.c: Sampling a Single Sky from a Point

$ raytraverse -c rayt.cfg sourcerun images

27-Jun-2022 12:58:14 SamplerArea Started sampling office at vp1 with may3pm
27-Jun-2022 12:58:14 SamplerArea level shape samples rate
27-Jun-2022 12:58:14 SamplerArea 1 of 1 [1 1] 1 100.00%
27-Jun-2022 12:58:14 SrcSamplerPt Started sampling office at view_000000 with may3pm
27-Jun-2022 12:58:14 SrcSamplerPt level shape samples rate
27-Jun-2022 12:58:14 SrcSamplerPt 1 of 6 [64 32] 2048 100.00%
27-Jun-2022 12:58:14 SrcSamplerPt 4 of 6 [512 256] 1030 0.79%
27-Jun-2022 12:58:14 SrcSamplerPt 5 of 6 [1024 512] 941 0.18%
27-Jun-2022 12:58:14 SrcSamplerPt 6 of 6 [2048 1024] 768 0.04%
27-Jun-2022 12:58:14 SrcSamplerPt total sampling: - 7656 0.37%
27-Jun-2022 12:58:19 SamplerArea total sampling: - 1 100.00%
27-Jun-2022 12:58:19 Integrator Making Images for 1 view directions at 1 points under 1 skies
   Making Images (01 of 01): 100%|█████████████████████████████████████| 1/1
   [00:00-00:00, 1.30it/s]
default_may3pm-0000_pt-0.6_1.0_1.2_vd-1.0_0.0_0.0.hdr
Exercise #1.d: Command Line Options: Interpolation

Naive Interpolation
~9 sec (sampling + interpolation)

Context Filtering
~14 sec (sampling + interpolation)

Context Sampling + higher accuracy
~2.5 min (sampling + interpolation)

raytraverse -c rayt.cfg images -basename inter -interpolate high
draytraverse -c rayt.cfg images -basename context -interpolate highc
raytraverse -c rayt.cfg sourceengine -accuracy .125 sourcerun --overwrite --scenedetail images\ -basename detail -interpolate high
setting up a simulation
Scene

[raytraverse_scene]
log = True
out = None
overwrite = False
reload = True
scene = None
[raytraverse_area]
  jitterrate = 0.5
  name = plan
  printdata = False
  printlevel = None
  ptres = 1.0
  rotation = 0.0
  static_points = None
  zheight = None
  zone = None
[raytraverse_suns]
epwloc = False
jitterrate = 0.5
loc = None
name = suns
printdata = False
printlevel = None
skyro = 0.0
sunres = 9

--no-epwloc // epwloc = False

--epwloc // epwloc = True
Suns

--no-epwloc // epwloc = False

--epwloc // epwloc = True
[raytraverse_skydata]
ground_fac = 0.2
loc = None
minalt = 2.0
mindiff = 5.0
mindir = 0.0
name = skydata
printdata = False
printfull = False
reload = True
skyres = 15
skyro = 0.0
wea = None
Available Methods

- **1comp**: daylight coefficient method, solar energy in sky patch
- **1compdv**: 1comp, but with direct view replacement of sun and specular reflections
- **2comp**: sky patch for sky contribution, sun run for sun contribution, depth of contributions depends on skyengine and sunengine settings, no approximation for sun from sky patch
- **3comp**: 2-phase DDS, sky handles sky+indirect sun, sun handles direct sun requires directskyrun -ab 1 and sunrun -ab 0
- **<sourcename>**: use a single source file (electric lights, single sky + sun, etc...)
Sky Patch Coefficients

1comp

1compdv
Sky Patch + Sun

2comp
Sky Patch + Direct Sun

3comp
Single Source

may3pm

sep2pm
## Simulation Types

<table>
<thead>
<tr>
<th></th>
<th>1comp</th>
<th>1compdv</th>
<th>2comp</th>
<th>3comp</th>
<th>&lt;srcname&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Run:</strong></td>
<td>skyrun</td>
<td>skyrun</td>
<td>skyrun</td>
<td>skyrun</td>
<td>sourcerun</td>
</tr>
<tr>
<td></td>
<td>directskyrun</td>
<td>sunrun</td>
<td>directskyrun</td>
<td>sunrun</td>
<td></td>
</tr>
<tr>
<td><strong>Config:</strong></td>
<td>[raytraverse_images]</td>
<td>[raytraverse_images]</td>
<td>[raytraverse_images]</td>
<td>[raytraverse_images]</td>
<td>[raytraverse_images]</td>
</tr>
<tr>
<td></td>
<td>simtype = 1comp</td>
<td>dcompargs = -ab 0</td>
<td>rayargs = -ab ACTUAL</td>
<td>dcompargs = -ab 1</td>
<td>simtype = srcname</td>
</tr>
<tr>
<td></td>
<td>[raytraverse_evaluate]</td>
<td>[raytraverse_images]</td>
<td>simtype = 2comp</td>
<td>simtype = 3comp</td>
<td>[raytraverse_evaluate]</td>
</tr>
<tr>
<td></td>
<td>simtype = 1comp</td>
<td></td>
<td></td>
<td></td>
<td>simtype = srcname</td>
</tr>
<tr>
<td></td>
<td>[raytraverse_evaluate]</td>
<td>[raytraverse_evaluate]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>simtype = 1comp</td>
<td>simtype = 2comp</td>
<td>simtype = 3comp</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
exercise #2
exercise #2: Annual Metrics From a Point

[shared]
wea = ../scene/geneva.epw

dimensions: 5.95 m x 4.60 m x 2.65 m

[raytraverse_scene]
out = office
scene = ../scene/office_glz.rad

[raytraverse_area]
name = vp1
static_points = 0.6,1,1.2

dimensions: 50% VLR

[raytraverse_suns]
epwloc = True
loc = ${shared:wea}

[raytraverse_skydata]
wea = ${shared:wea}

[raytraverse_skyengine]
dcompargs = -ab 0

[raytraverse_evaluate]
basename = results
blursun = True
metrics = illum dgp
resampleview = True
sdirs = 1,0,0 0,-1,0
sensors = 0.6,1,1.2
simtype = 1compdv
threshold = 2000.0

[raytraverse_pull]
col = view metric
gridhdr = True
lr = results.npz
metricfilter = dgp
ofiles = dgp
skyfill = office/skydata.npz
Exercise #2.a: Annual Metrics From a Point

# run a 1compdv simulation from a single point,
# evaluate DGP and Illum for 2 view directions,
# create hdr heatmaps

raytraverse -c rayt.cfg skyrun directskyrun evaluate pull
Exercise #2.b: Using Pull

# inspect result object:
raytraverse pull -lr results.npz --info

# create tsv txt files for each view:
raytraverse pull -lr results.npz -col 'metric view' -ofiles test

# pad to 8760 data:
raytraverse pull -lr results.npz -col 'metric view' -ofiles test8760 -skyfill office/skydata.npz

# look at 12pm each day for 1 view:
raytraverse pull -lr results.npz -viewfilter 0 -skyfilter 12:8760:12 -skyfill office/skydata.npz

# calculate percent of daylight hours with perceptible glare for both views:
raytraverse pull -lr results.npz -metricfilter dgp --no-header --no-rowlabel -col view \
  | rcalc -e '$1=if($1-.35,1,0);$2=if($2-.35,1,0)$' | total -m
simulation parameters
Daylight Coefficients / Sky Simulation

[raytraverse_area]
jitter_rate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_skyengine]
accuracy = 1.0
dcompargs = -ab 1
default_args = True
idres = 32
nlev = 5
rayargs = None
skyres = 15
vlt = 0.64

[raytraverse_skyrun]
accuracy = 1.0
edgemode = reflect
jitter = True
nlev = 3

Zone

Static Points
Daylight Coefficients / Sky Simulation

[raytraverse_area]
jitterrate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_skyengine]
accuracy = 1.0
dcompargs = -ab 1
default_args = True
idres = 32
nlev = 5
rayargs = None
skyres = 15
vlt = 0.64

[raytraverse_skyrun]
accuracy = 1.0
edgemode = reflect
jitter = True
nlev = 3
Sun Coefficients Simulation

[raytraverse_area]
jitterrate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_suns]
jitterrate = 0.5
sunres = 9

[raytraverse_sunengine]
accuracy = 1.0
default_args = True
idres = 32
nlev = 6
rayargs = -ab 0
vlt = 0.64

[raytraverse_sunrun]
accuracy = 1.0
edgemode = reflect
jitter = True
nlev = 3
srcaccuracy = 1.0
srcjitter = True
srcnlev = 3
Source Simulation / Scene Detail

```
[raytraverse_area]
jitterrate = 0.5
ptres = 1.0
static_points = None
zone = None

[raytraverse_sourceengine]
accuracy = 1.0
color = True
default_args = True
idres = 32
nlev = 6
rayargs = None
vlt = 1.0

[raytraverse_sourcerun]
accuracy = 1.0
distance = 0.5
edgemode = reflect
jitter = True
nlev = 3
normal = 5.0
scenedetail = False
```
basic scripting
Setup

```python
from raytraverse.scene import Scene
from raytraverse.mapper import PlanMapper, SkyMapper
from raytraverse.sky import SkyData

def setup():
    scn = Scene("room", "room.rad")
    pm = PlanMapper("plane.rad")
    skyd = SkyData("weather.epw")
    skyd.write(scene=scn)
    sm = SkyMapper(skyd.loc)
    return scn, pm, sm, skyd
```

```
raytraverse scene --out room --scene room.rad \
area --zone plane.rad \
skydata --wea weather.epw \
suns --loc weather.epw --epwloc
```

Reload:

```python
from raytraverse import api

scn, pm, skyd = api.auto_reload("room", "plane.rad")
```

raytraverse.readthedocs.io/en/stable/scene.html#scene

raytraverse.readthedocs.io/en/stable/mapper.html#planmapper

raytraverse.readthedocs.io/en/stable/sky.html#skydata

from raytraverse.sampler import ISamplerArea, ISamplerSuns, Sensor
from raytraverse.renderer import Rcontrib, Rtrace

def sample(scn, pm, sm):
    rcontrib = Rcontrib("-I+", scene=scn.scene)
    sensor = Sensor(rcontrib)
    sampler = ISamplerArea(scn, sensor, stype='sky')
    skylp = sampler.run(pm)
    
    Rcontrib.reset()
    Rcontrib("-I+ -ab 1", scene=scn.scene)
    skydlp = sampler.repeat(skylp, 'skydcomp')
    Rcontrib.reset()
    
    rtrace = Rtrace("-I+ -ab 0", scene=scn.scene)
    sensor = Sensor(rtrace)
    sampler = ISamplerSuns(scn, sensor)
    sunlp = sampler.run(sm, pm)
    Rtrace.reset()
from raytraverse import api

from raytraverse.integrator import SensorIntegrator
from raytraverse.lightfield import SensorPlaneKD, SunSensorPlaneKD

scn, pm, skyd = api.auto_reload("office", ".//scene/plane.rad")

skylp = SensorPlaneKD(scn, None, pm, "sensor_sky")
skydmp = SensorPlaneKD(scn, None, pm, "sensor_skydmp")
sunlp = SunSensorPlaneKD(scn, None, pm, "sensor_suns_sun")

itg = SensorIntegrator(skylp, sunlp, skydmp,
                          ptype="skysun", "sun", "patch"), factors=(1, 1, -1))

zlr = itg.zonal_evaluate(skyd, pm)
zlr.write("results.npz")

itg = api.get_integrator(scn, pm, simtype="1comp")

lr = itg.evaluate(skyd, [2, 2, 1.2])
lr.write("result_22.npz")

Sensor based evaluation:
currently no command line

Full evaluation:
raytraverse evaluate
```python
import numpy as np
from raytraverse import api
from raytraverse.lightfield import ZonalLightResult, LightResult

scn, pm, skyd = api.auto_reload("office", "plane.rad", ptres=0.6)

lr = LightResult("sunresults_grid.npz")
skyfilter = skyd.masked_idx(np.arange(1896, 1920))
lr.pull2planhdr("plane.rad", "testg", sky=skyfilter)

zlr = ZonalLightResult("sunresults_full.npz")
d, l, n = zlr.pull("metric", metric=[zlr.axis("metric").index("illum")])
print(d.shape)  # (572262, 1)
lr2 = zlr.rebase(pm.point_grid(False, level=2))
d, l, n = lr2.pull("metric", metric=[lr2.axis("metric").index("illum")])
print(d.shape)  # (2179584, 1)
```

```
raytraverse pull \
-lr sunresults_grid.npz \
-skyfill office/skydata.npz \
-skyfilter 1896:1920 \
-ofiles pullg --gridhdr \
-imgzone plane.rad
```
exercise #3
exercise #3: Sensor Point Simulation

- Use Sensor points to sample illuminance in a space using 3 component method
- Synthesize illuminance plans for a range of dates
- Evaluate direct sun illuminance annually to compute ASE
- compare different interpolation resolutions with zonal analysis
exercise #3.a: sample

# run sensor sampler for 3 components and evaluate for mar 21st.
./sample.py

# use command line to pull to hdr images
raytraverse pull -lr results_0321.npz -imgzone plane.rad --gridhdr --ofiles plan \
-metricfilter "illum src_err src_idx area"
exercise #3.a: evaluate

# reload sampling results and run full annual evaluation of sun contributions
./evaluate.py

```
(dev39) Stephens-MacBook-Pro:ex3 stephenwasilewski$ raytraverse pull -lr sunresults_full.npz --info
ZonalLightResult sunresults_full.npz:
Has 4 axes: ['sky', 'zone', 'view', 'metric']
  Axis 'sky' has length 4257:
   0 (1., 1., 9.5)
   1 (1., 1., 10.5)
   2 (1., 1., 11.5)
   3 (1., 1., 12.5)
   4 (1., 1., 13.5)
...
   4252 (12., 31., 12.5)
   4253 (12., 31., 13.5)
   4254 (12., 31., 14.5)
   4255 (12., 31., 15.5)
   4256 (12., 31., 16.5)
  Axis 'zone' has length 1:
   0 plan
  Axis 'view' has length 1:
   0 (0., 0., 0., 0., 0., 1.)
  Axis 'metric' has length 5:
   0 x
   1 y
   2 z
   3 area
   4 illum
```
exercise #3.c: analyze

# use lightresult object to calculate ASE in different ways
./calc.py

# compare convergence of different interpolation levels on ASE
cl_plot scatter ase_interpolated.txt -x_vals 0 -y_vals 1:5 -labels "raw 2.0 1.0 0.5"
--legend -axes 'hours above,0,2000,fraction of floor area,0,.75' -ms 0 -lw '3 0' -ms '0 3'
-colors viridis_r -mew 0