Acceleration of Plume Tracker Volcanic Emission Analysis Software

A. Berk and C. S. Guiang*
Spectral Sciences, Inc.

V. J. Realmuto
Jet Propulsion Laboratory,
California Institute of Technology

JPL
Jet Propulsion Laboratory
California Institute of Technology

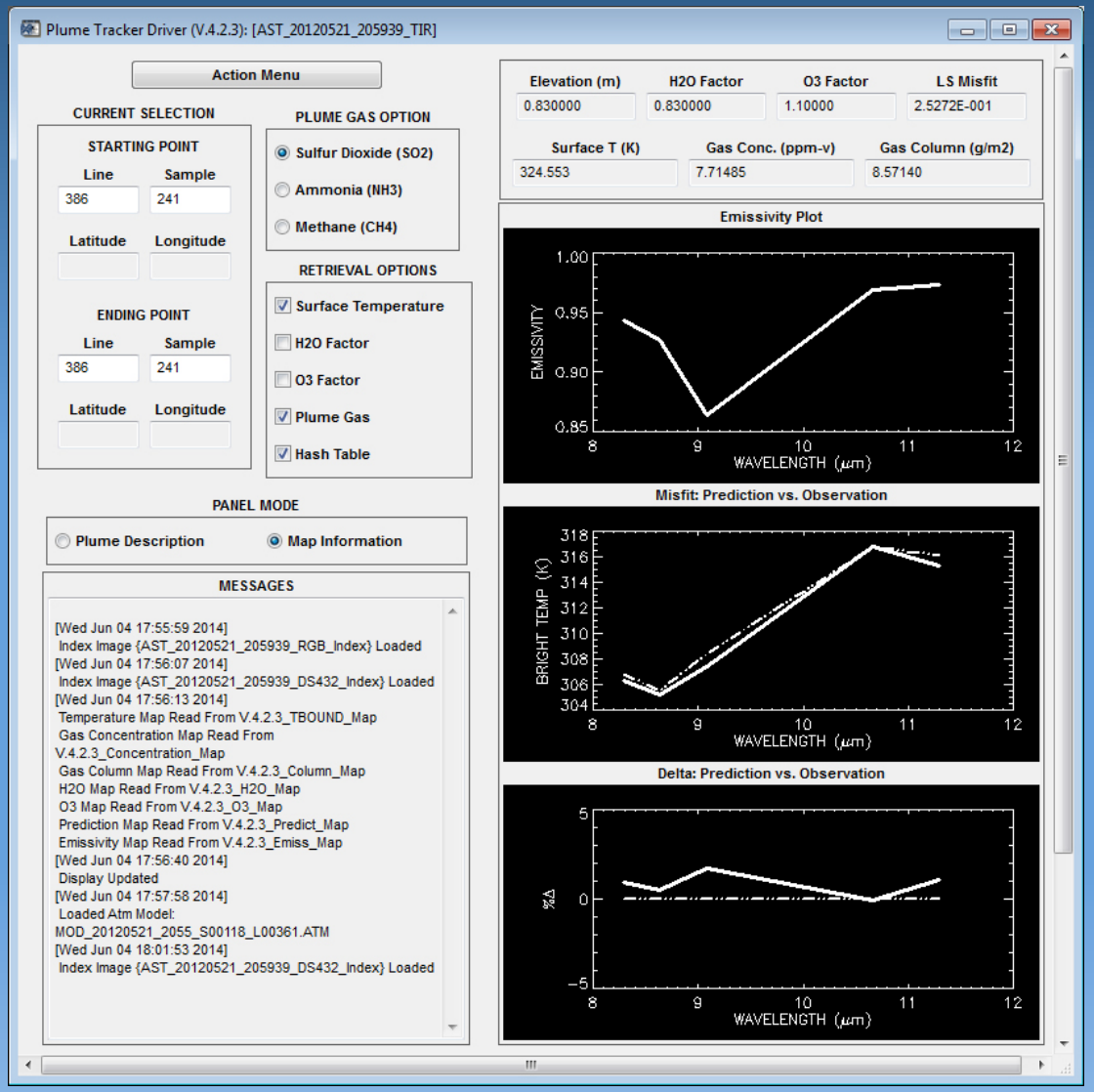
*chona@spectral.com

Outline

- What is Plume Tracker?
- Current Plume Tracker implementation
- Performance enhancements
  - Algorithmic improvements
  - Computational improvements
- Summary and future directions

Plume Tracker: Interactive Toolkit for Deep Analysis of TIR Image Data

1. Graphic User Interface
2. Radiative Transfer Model Based on MODTRAN®
3. Retrieval Procedures
   - Ground temperature and emissivity
   - H_2O Vapor and O_3 Scaling Factors
   - SO_2, NH_3, and CH_4 concentrations
   - Optimized for Two-component retrievals (Temperature + Gas Concentration)



Plume Tracker Driver Widget

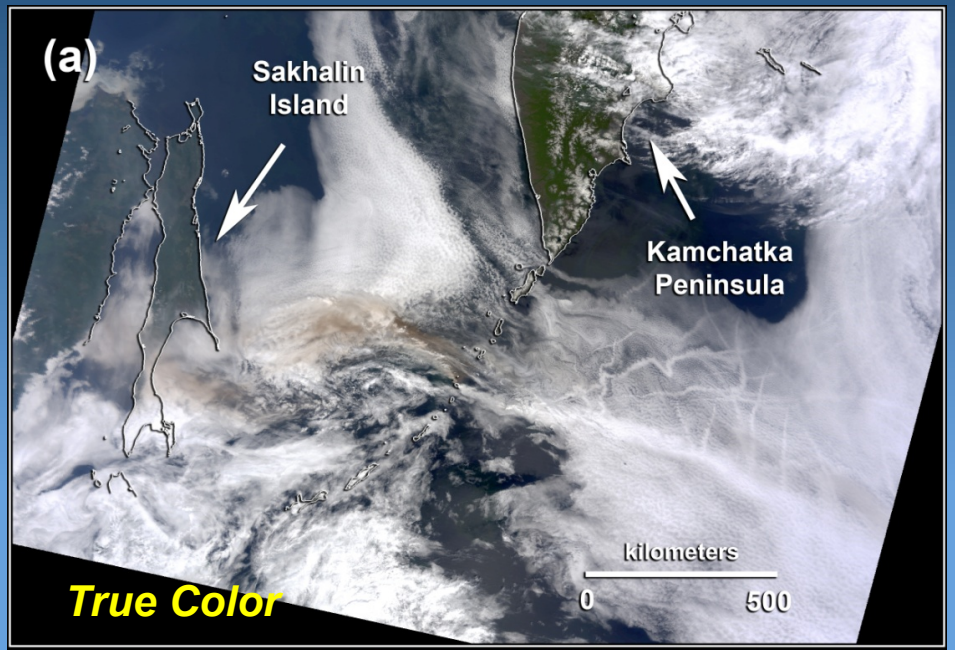
Detection of Volcanic Plumes in the Thermal Infrared (TIR)

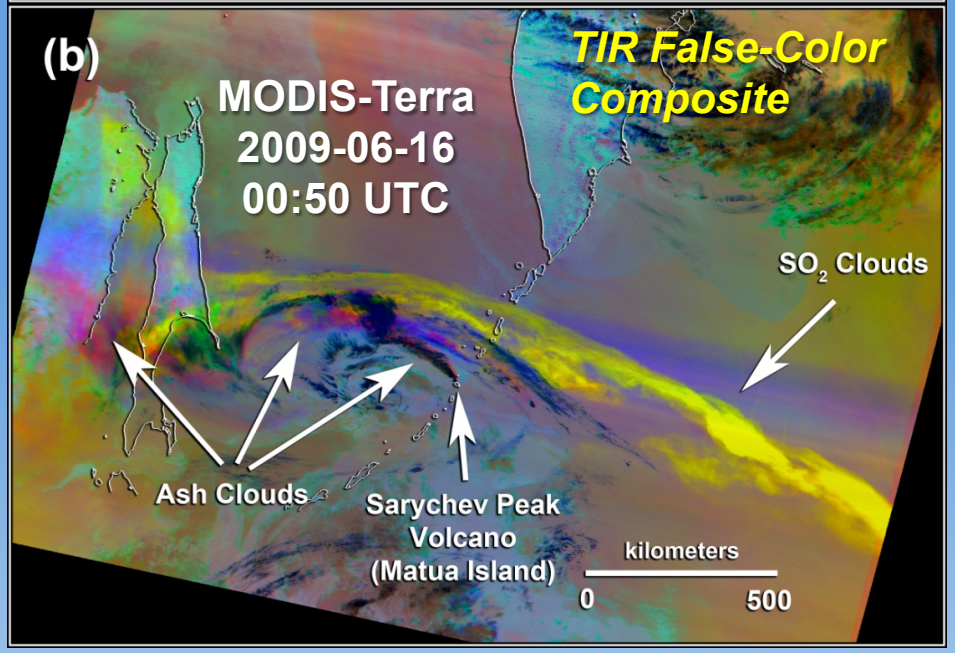
(a) True-Color Composite

- Difficult to Discriminate Volcanic Clouds from Meteorological (Met) Clouds
- Recognition of Ash Clouds Requires Fairly High Mass Loads
- SO_2 Clouds are Invisible

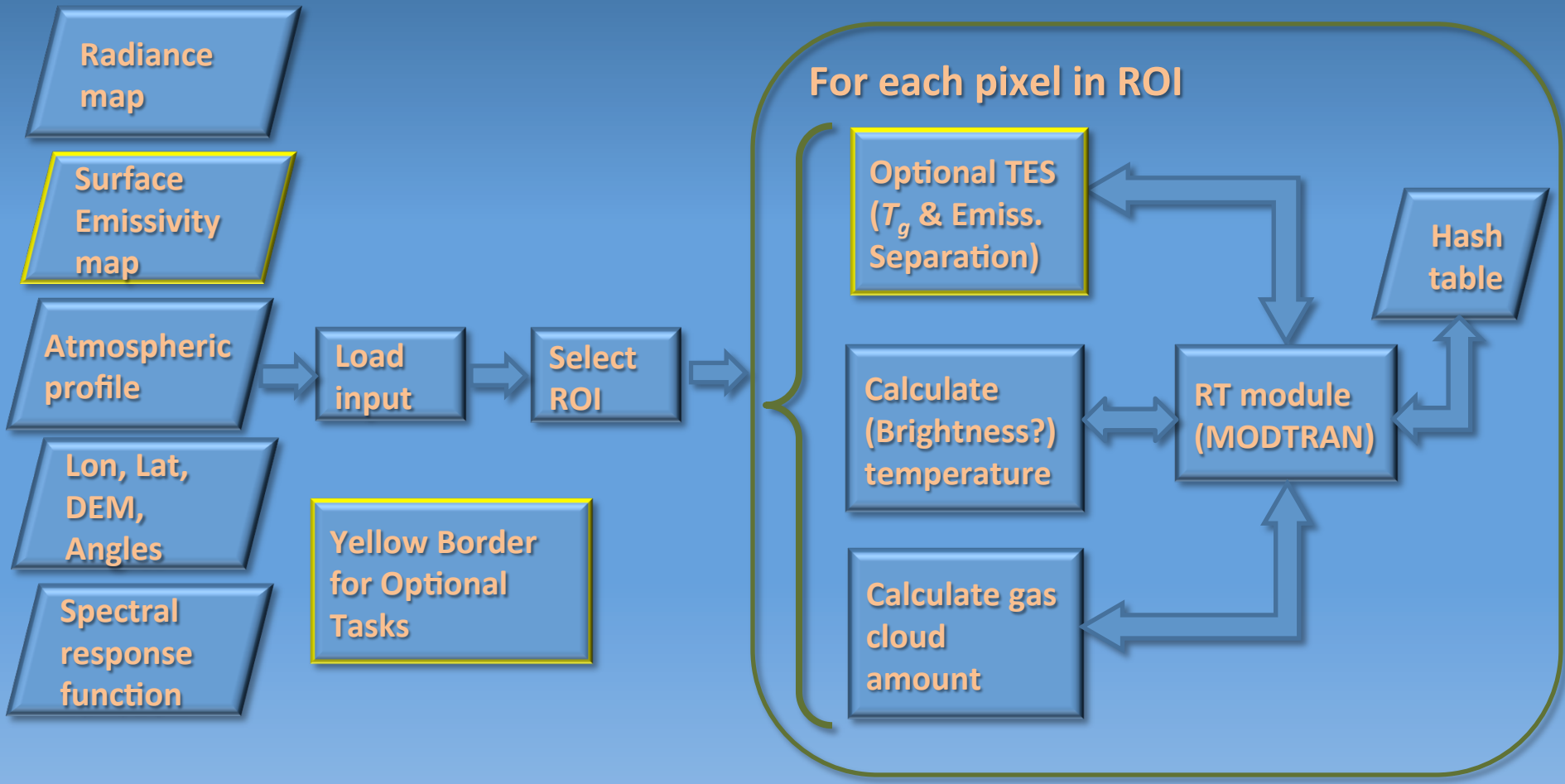
(b) TIR False-Color Composite

- Met Clouds and Volcanic Products are Easy to Discriminate in TIR
- Quantification of Ash or SO_2 Content is Challenging!



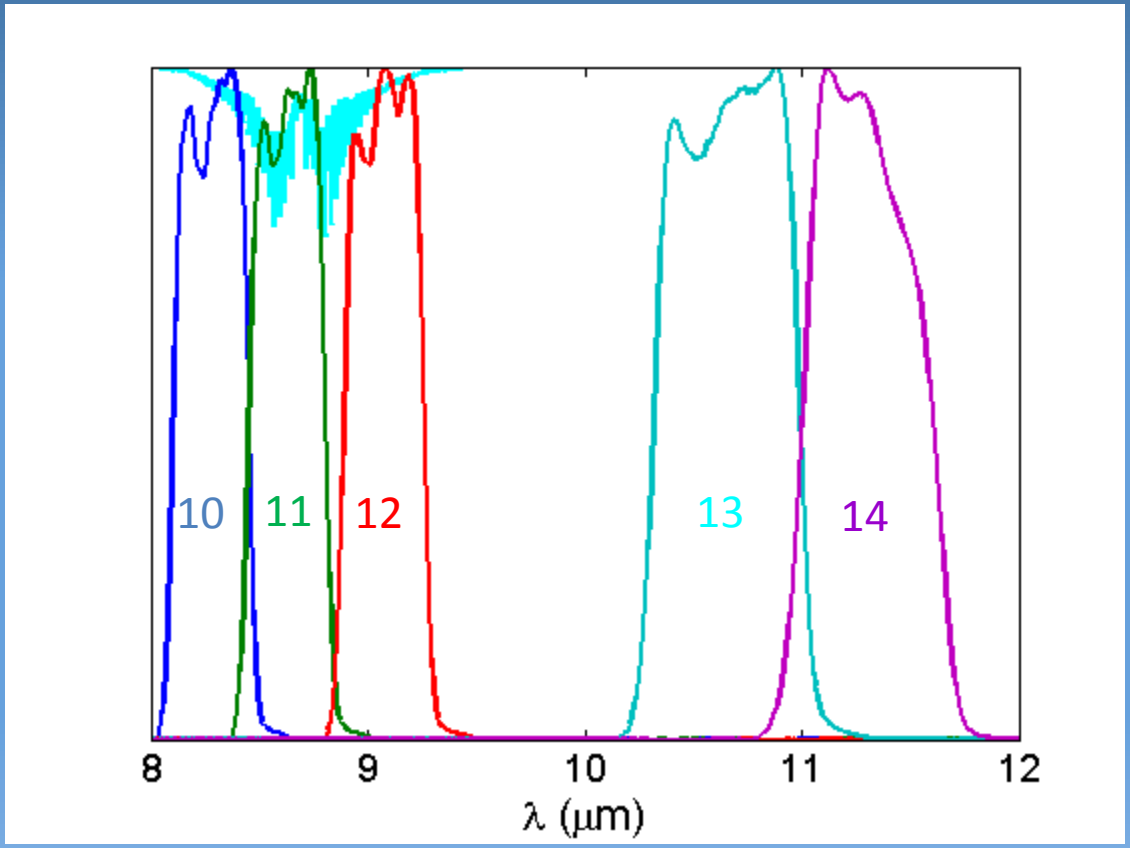


Plume Tracker processing flow



Current production version in IDL/C++/Fortran
Converting to C++/Fortran

SO_2 IR Spectra

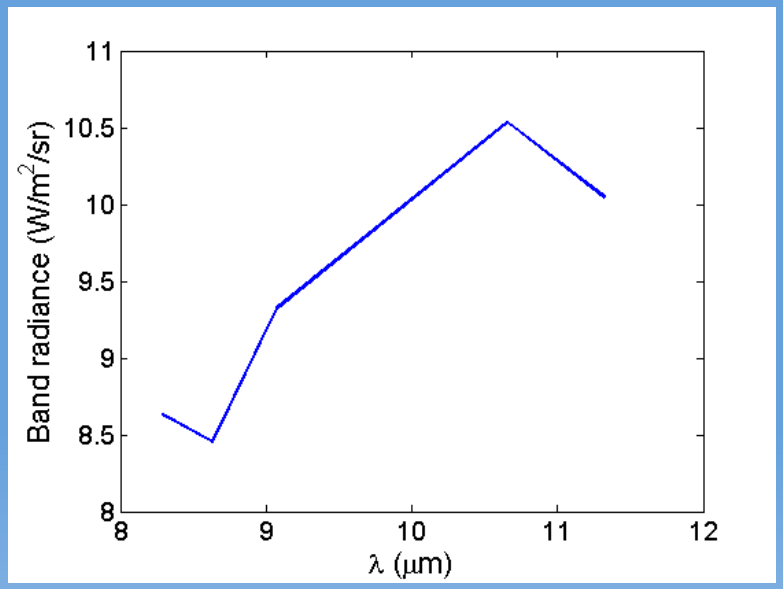


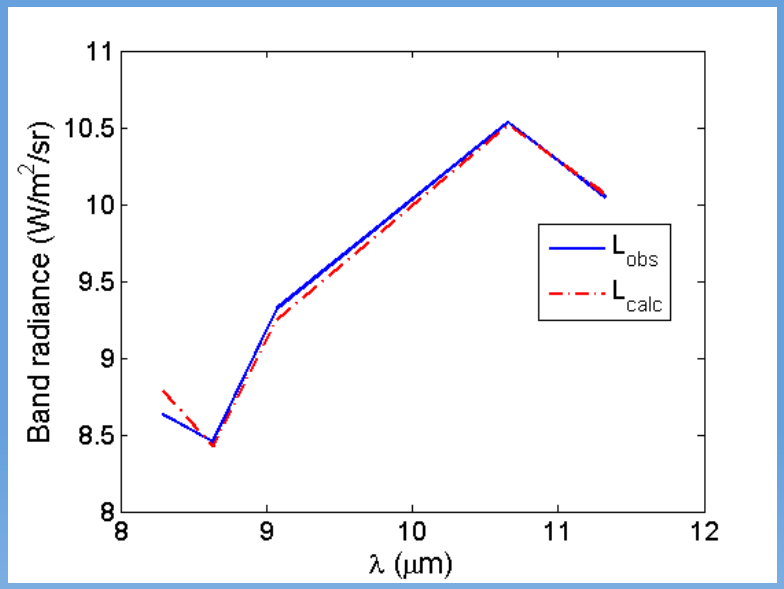
SO_2 IR absorption plotted over ASTER IR bands 10-14

Retrieval of ground temperature and SO_2

Given TIR spectra L_{obs}, ground altitude, sensor zenith angle

Find $[H_2O]$ and $[O_3]$ column amounts, and T_g and $[SO_2]$ maps that yield closest matching radiance spectra L_{calc} to observed radiance



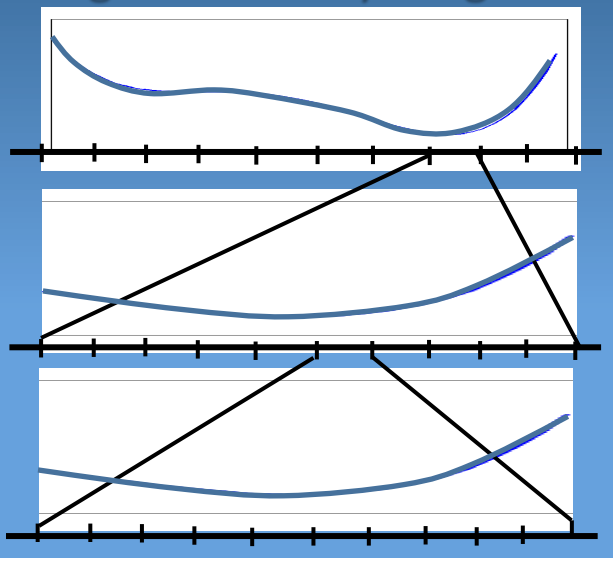


Earlier retrieval approaches

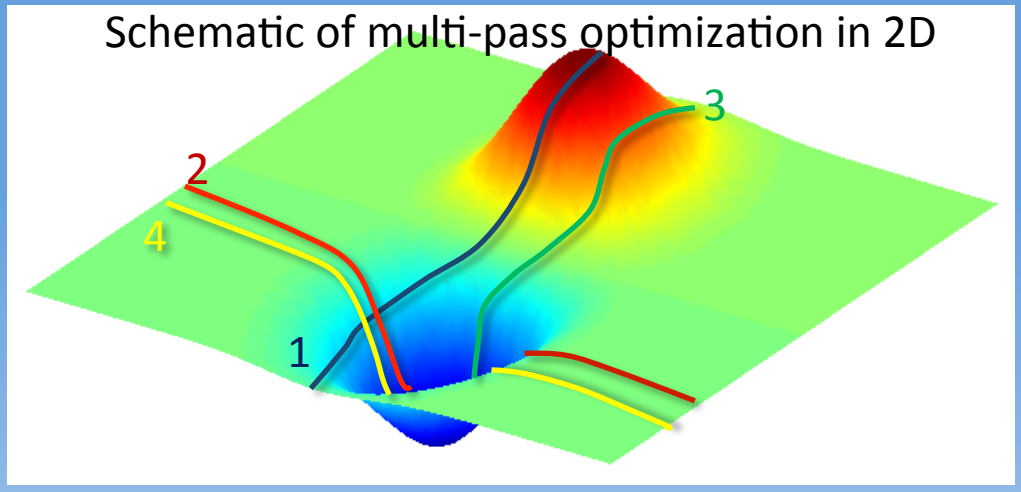
- Domain Refinement
  - Divide T_g, $[H_2O]$, $[O_3]$, $[SO_2]$ into 1D grids
  - Find minimum interval in T_g. Refine grid, then find minimum again. Repeat for a total of three passes.
  - Perform similar cycle for gases.

- Multi-pass optimization
  - Perform 1D optimizations over T_g then gases
  - Repeat until successive iterations do not improve T_g estimate.

Bracketing followed by 1D grid refinement



Schematic of multi-pass optimization in 2D



MODTRAN is run for each set of {T_g, $[SO_2]$, etc.} input

Algorithmic improvements – Hash table

- MODTRAN calculation is time-intensive; why not save individual runs?

Key(1)	MODTRAN Spectrum(1)
Key(2)	MODTRAN Spectrum(2)
Key(3)	MODTRAN Spectrum(3)
Key(4)	MODTRAN Spectrum(4)
…	…

Key(i) = [*zenith angle*][*surface elevation*][*surface temp*] [H_2O *factor*][O_3 *factor*][SO_2 *factor*]

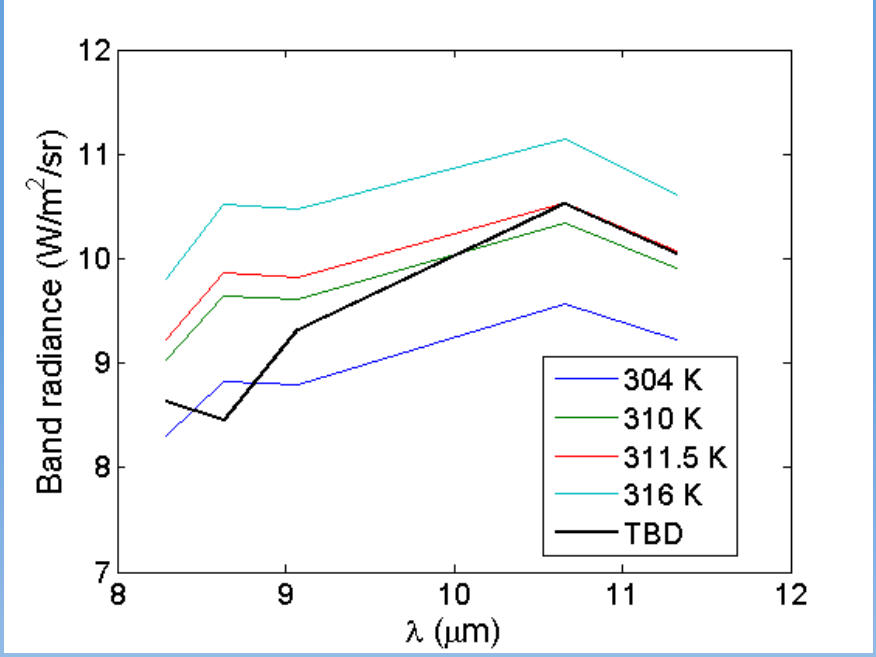
  - Discretize each input parameter, and construct key as a function of bin indices.
  - Unlike lookup table, no need to run MODTRAN for each point in discrete parameter space

Algorithmic improvements – Reconstructed radiance

- Atmospheric Up-welling radiance (U), Down-welling radiance (D), and transmittance (*t*) are independent of ground temperature and emissivity
- Run MODTRAN once for *t*, U, D, and reconstruct L_{calc} for each temperature (T_g) and emissivity (*e*):

$$L_{calc} = U + e\,B(T_g)\,t + D\,(1-e)$$

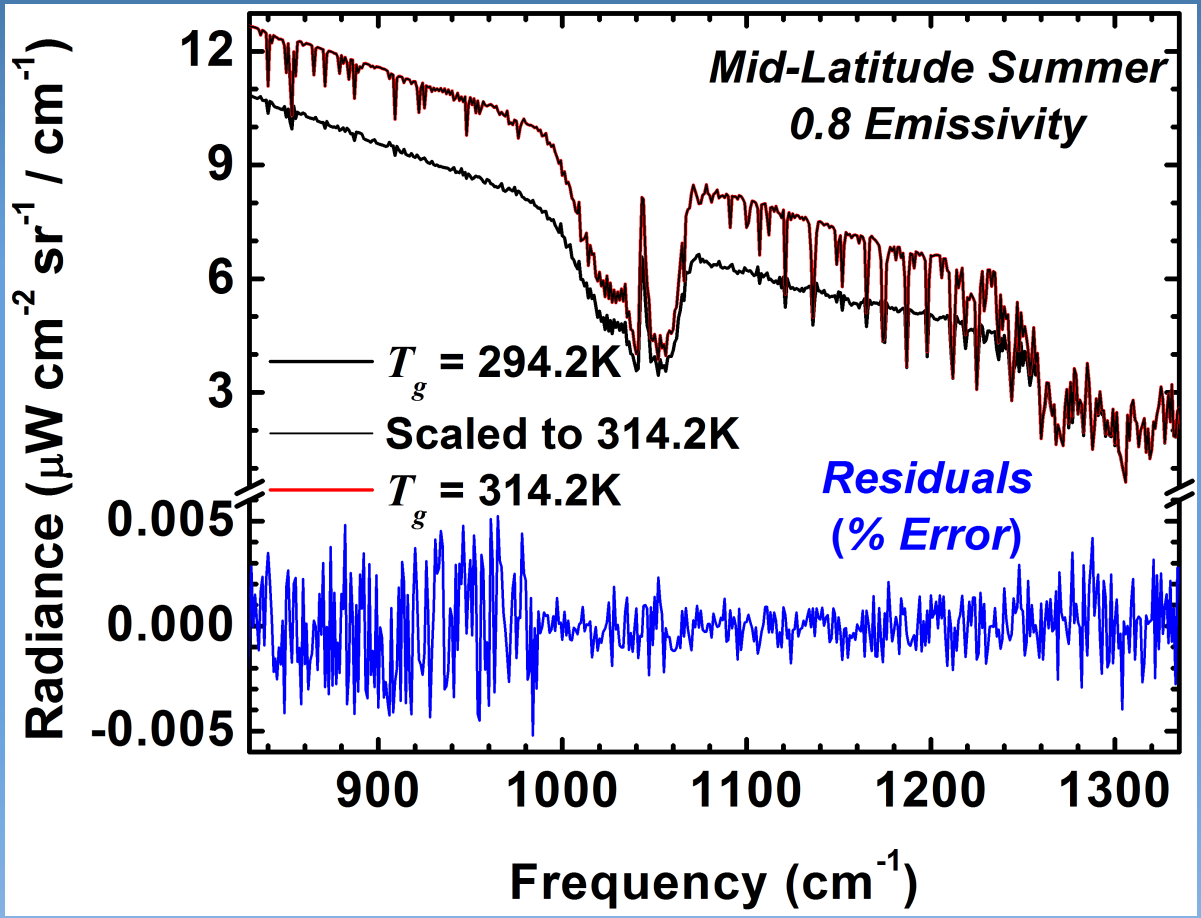
$$= \text{Upwelling} + \text{Surface Emission} + \text{Transmitted Reflected Downwelling}$$



- Cache *t*, U, D in hash table.
- Re-calculate *t*, U, D for changes in plume height, plume thickness, surface elevation, sensor zenith, or gas concentration.
- Removing T_g from hash key increases hash table utilization.

Find T_g such that L_{calc} bounds L_{obs} from above

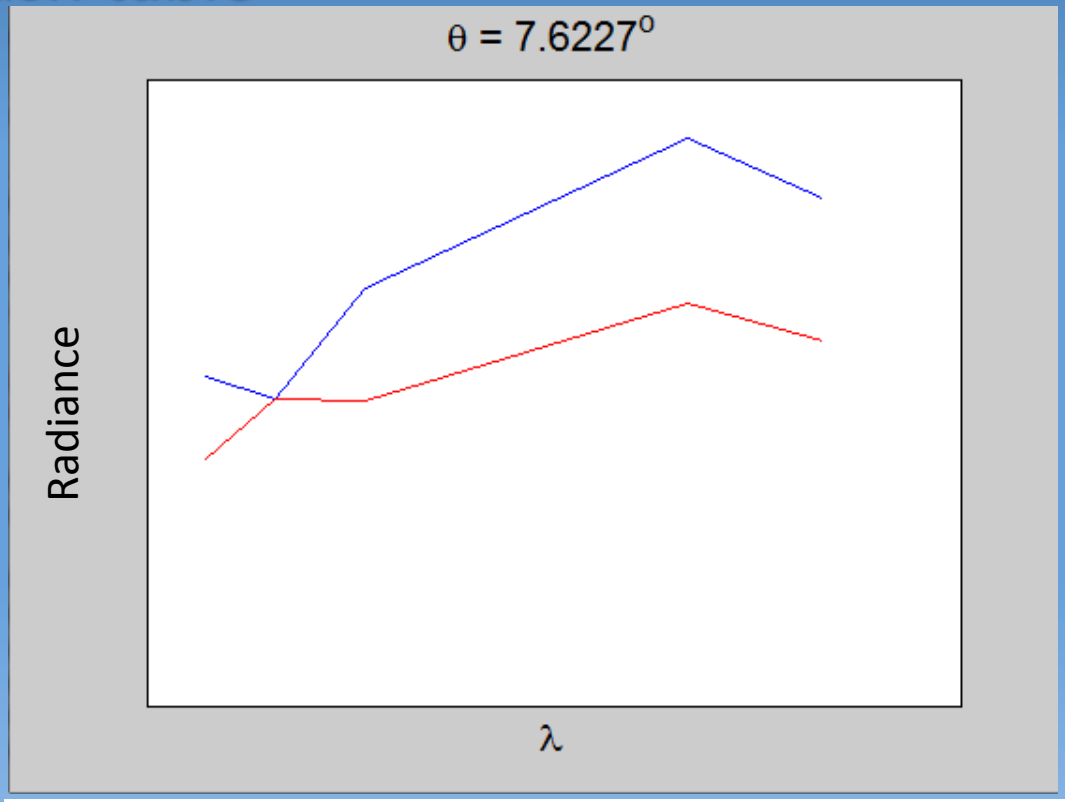
Reconstructed radiance --- is this a valid approach?



Yes!

Algorithmic improvements

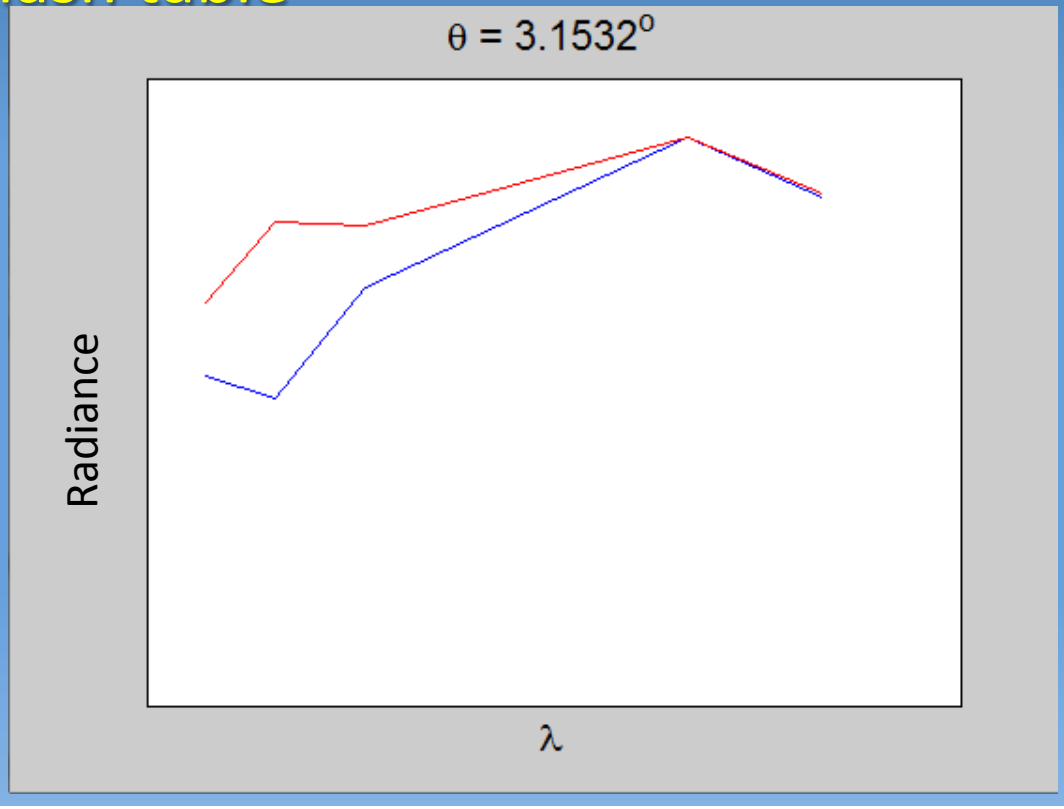
- Run Brent optimization over T_g, using scaling and hash table



$$\theta = \cos^{-1}\left(\frac{\mathbf{L}_{calc}\,\mathbf{g}\mathbf{L}_{obs}}{\left|\mathbf{L}_{calc}\right|\left|\mathbf{L}_{obs}\right|}\right) + \sum_{\lambda} H\left(\left(L_{obs} - L_{calc}\right)_{\lambda}\right)$$

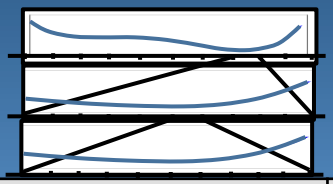
Algorithmic improvements

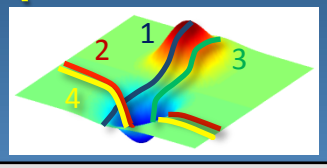
- Then, run Brent optimization over $[SO_2]$, also using hash table



$$\theta = \cos^{-1}\left(\frac{\mathbf{L}_{calc}\,\mathbf{g}\mathbf{L}_{obs}}{|\mathbf{L}_{calc}||\mathbf{L}_{obs}|}\right) + \sum_{\lambda} H\left(\left(L_{obs} - L_{calc}\right)_{\lambda}\right)$$

Retrieval Algorithm Performance (2011 – 2014)



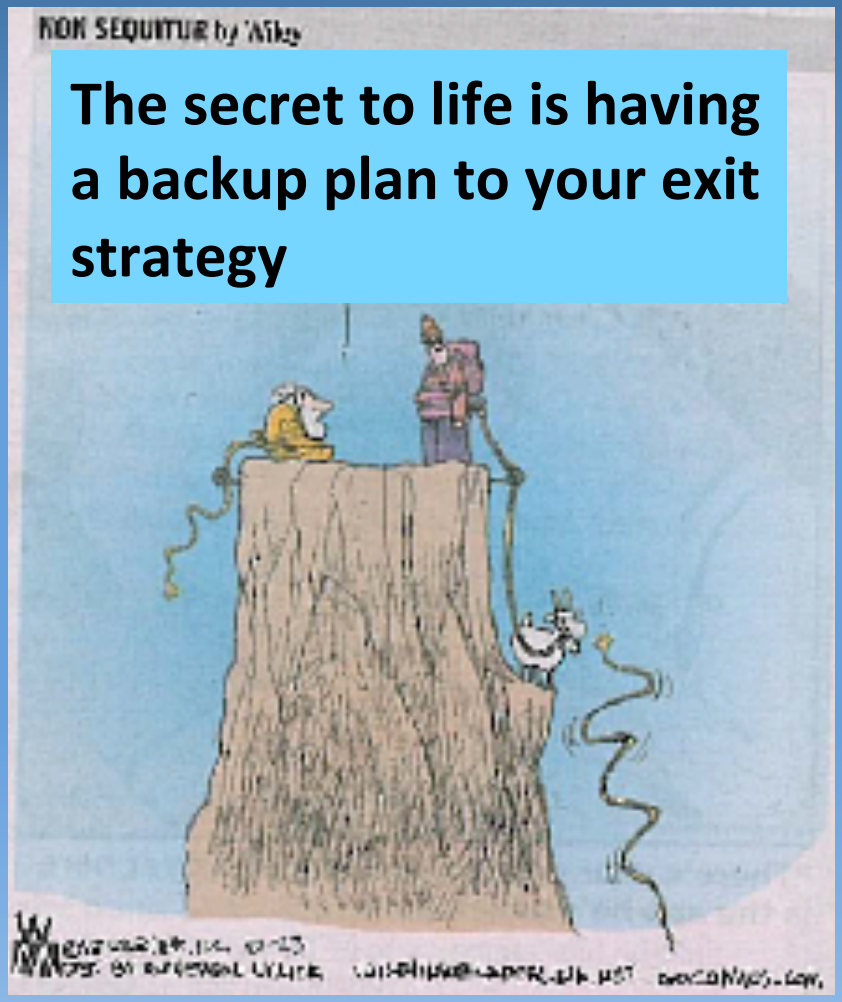


$$L_{calc} = U + e\,B(T)t + D\,(1 - e)$$

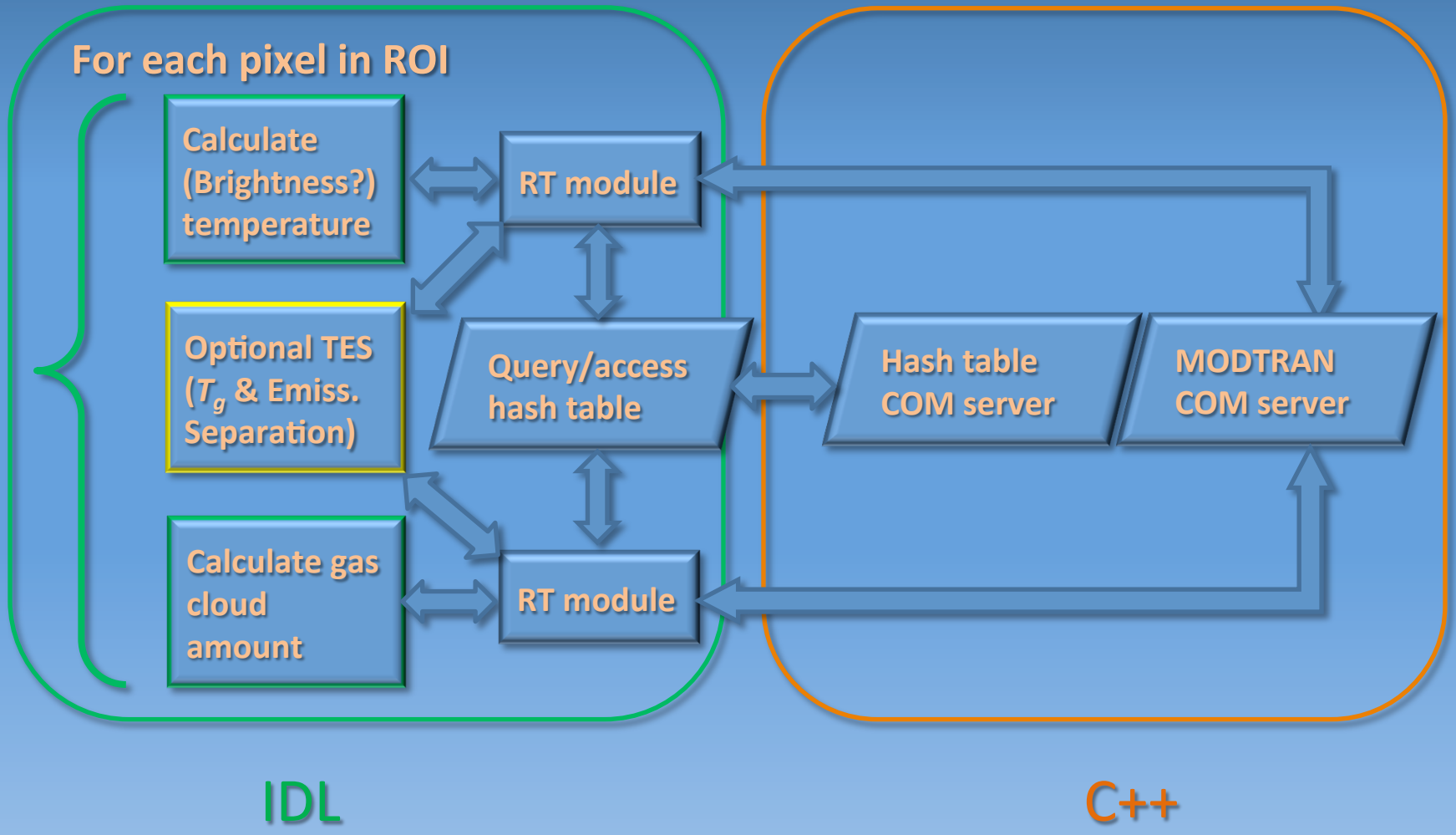
	Domain Refinement *(V.2.2.9)*	Multi-Pass Brent Minimization *(V.3.0)*	Single-Pass Brent Minimization *(V.3.2)*	Reconstructed Radiance *(V.4.2b)*
Calls to RT Model *ROI = 1296 pixels*	*264,398 (204 calls/pix)*	*144,508 (112 calls/pix)*	*58,361 (45 calls/pix)*	*29,485 (23 calls/pix)*
No Hash Table	*6.4 sec/pix*	*3.2 sec/pix*	*1.26 sec/pix*	*0.61 sec/pix*
Hash Table	*1.6 sec/pix*	*0.8 sec/pix*	*0.16 sec/pix*	*0.019 sec/pix*
Success Rate *Hash Table Utilization*	*77.4 %*	*77.0%*	*88.0%*	*97.8%*

Fundamental Issues with MODTRAN GPU Implementation

- MODTRAN3 is too fast!
  - Entire 8-12 μm bandpass runs in 0.04 sec
  - The bottleneck cumulative path transmittances are not independent
- GPUs have been used to speed up line-by-line (LBL) radiative transfer, but
  - LBL transmittances much more amenable to a GPU implementation than MODTRAN
  - LBL 8-12 μm thermal emission serial run 2-3 orders of magnitude slower
  - Adding scattering to LBL increases CPU processing time by an additional two orders of magnitude
- Back up plan: accelerate based on
  - Radiative transfer physics
  - Multi-threading



Plume Tracker v. 4.2b



Computational Optimization of PlumeTracker

- Port entire pixel retrieval from IDL to C++
  - Earlier version requires repeated calls to external functions *for* MODTRAN set-up, run and data retrievals, and *for* creation and update of hash table
    - Frequent data transfers consume additional time
    - Prone to memory allocation errors
  - Numerically intensive operations like convolution and interpolation are faster in compiled language
- Parallelize pixel retrieval
  - Takes advantage of multicore CPU-based hardware
  - Could lead to more efficient data access

C++ implementation of temperature and SO_2 retrievals

- Portable, no longer uses Microsoft-based COM technology
- C++11-compliant
- Single MODTRAN object encapsulates input variables, default parameters, run function and output post-processing
  - Uniform API for MODTRAN to allow for seamless transition from MODTRAN 3 to MODTRAN 5
- Special STL-based containers
  - Fortran-like access to 2D and 3D arrays
- Hash table based on **`unordered_map<long,vector<float>>`**
  - Key type is long, but can be changed to **`vector<short>`** if number of input parameters increases; will need new hash function
  - Use of **`unordered_map<keyType,returnType>`** makes it easy to switch key type

Old versus new hash table

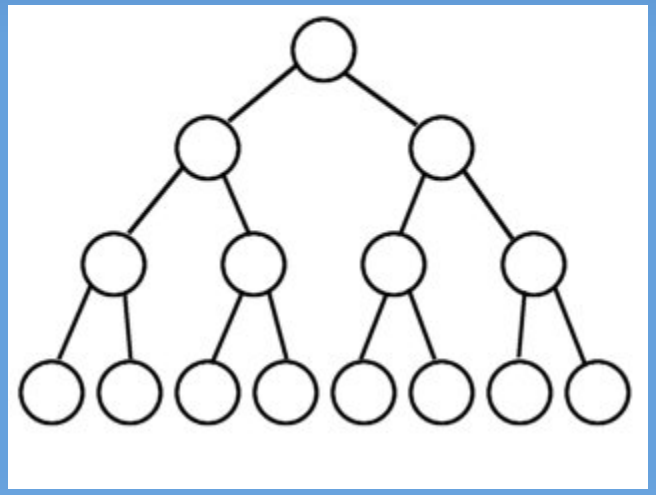
k_1	$\mathbf{t}_1$
k_2	$\mathbf{t}_2$
k_3	$\mathbf{t}_3$
•	•
•	•
•	•
k_n	$\mathbf{t}_n$

k_1	$\mathbf{U}_1$
k_2	$\mathbf{U}_2$
k_3	$\mathbf{U}_3$
•	•
•	•
•	•
k_n	$\mathbf{U}_n$

k_1	$\mathbf{D}_1$
k_2	$\mathbf{D}_2$
k_3	$\mathbf{D}_3$
•	•
•	•
•	•
k_n	$\mathbf{D}_n$

3 separate $O(N)$ searches

```
std::unordered_map<long,std::vector<T>>
```



$O(\log N)$ search

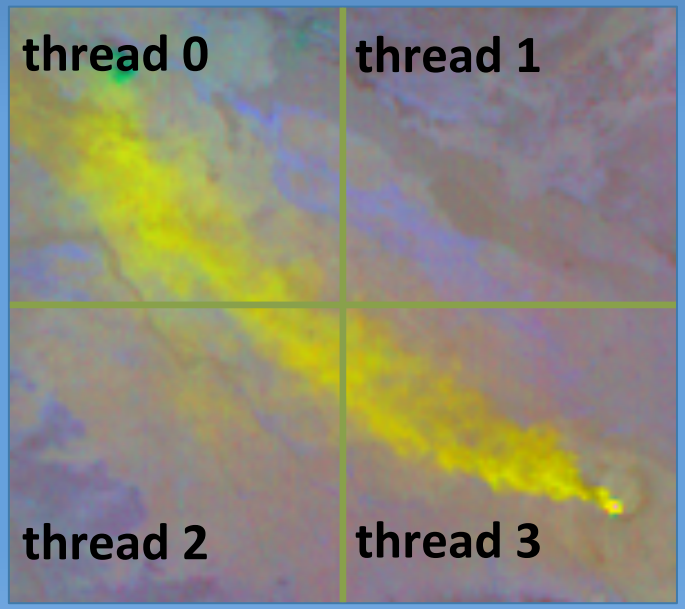
Additional optimizations

- "Convolution" of *e*,  with sensor spectral response is performed in C++
- Interpolation, Brent, etc. also in C++
- Single call to C++ function

Multithreaded parallelization of pixel retrieval

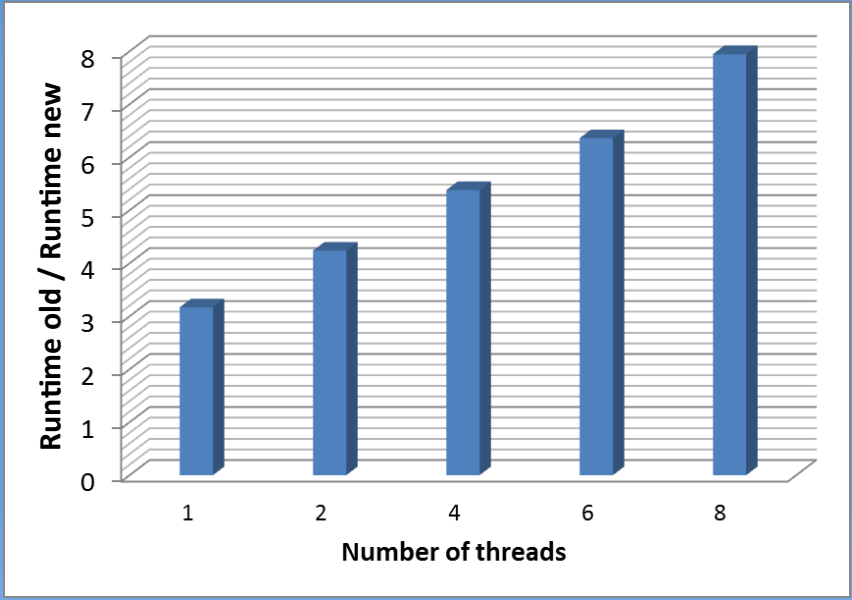
Strategy

- Divide ROI among physical number of cores
- Use OpenMP to perform retrievals in parallel, distributing pixels among threads
- Speedup scales linearly with thread count



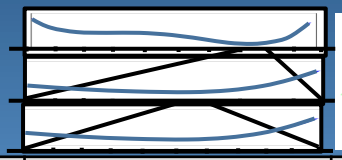
Region of Interest (ROI)

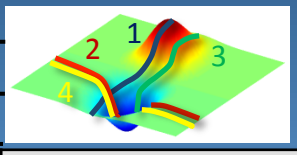
Performance relative to IDL version



- Serial C++ implementation delivers 3x faster retrieval
- Use of 8 threads increases performance to ~8x
- Scaling limited by hash table
  - Read/write access to hash table is serialized
  - For thread-safety, simultaneous read and write to hash table needs to be prevented;
  - However, `omp_get_lock`/ `omp_set_lock` mechanism also prevents simultaneous reads, which are thread-safe
- Still need to evaluate effect of thread scheduling on speed

Retrieval Algorithm Performance (2011 – 2014)





$$L_{calc} = U + e\,B(T)t + D\,(1 - e)$$

	Domain Refinement *(V.2.2.9)*	Multi-Pass Brent Minimization *(V.3.0)*	Single-Pass Brent Minimization *(V.3.2)*	Reconstructed Radiance *(V.4.2b)*	Parallel, C++
Calls to RT Model *ROI = 1296 pixels*	*264,398 (204 calls/pix)*	*144,508 (112 calls/pix)*	*58,361 (45 calls/pix)*	*29,485 (23 calls/pix)*	
No Hash Table	*6.4 sec/pix*	*3.2 sec/pix*	*1.26 sec/pix*	*0.61 sec/pix*	
Hash Table	*1.6 sec/pix*	*0.8 sec/pix*	*0.16 sec/pix*	*0.019 sec/pix*	*0.0021 sec/pix*
Success Rate *Hash Table Utilization*	*77.4 %*	*77.0%*	*88.0%*	*97.8%*	*97.8%*

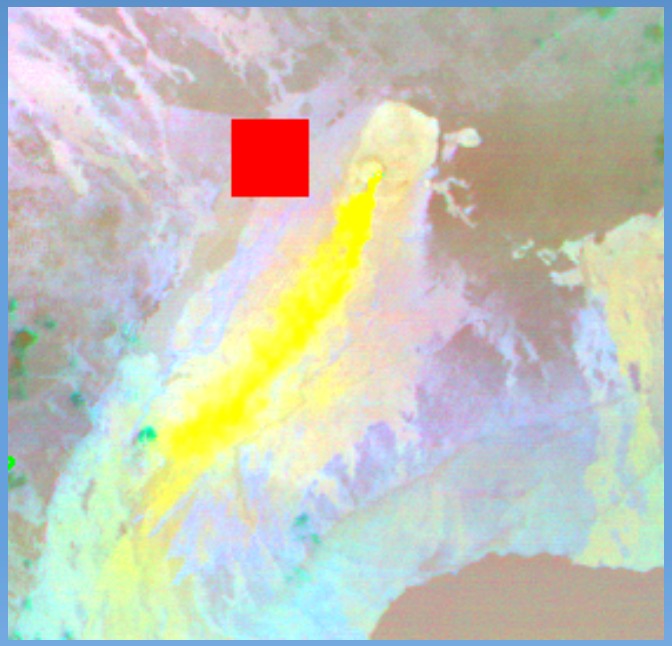
Summary

- Algorithmic improvements to retrievals led to significant speedups
  - Caching MODTRAN runs in hash table is particularly efficient
  - Reconstructed radiance by scaling T_g led to 2x fewer MODTRAN evaluations
- Port to C++ by itself improved performance by 3x
- Some room for improvement in parallel performance

Backup slides

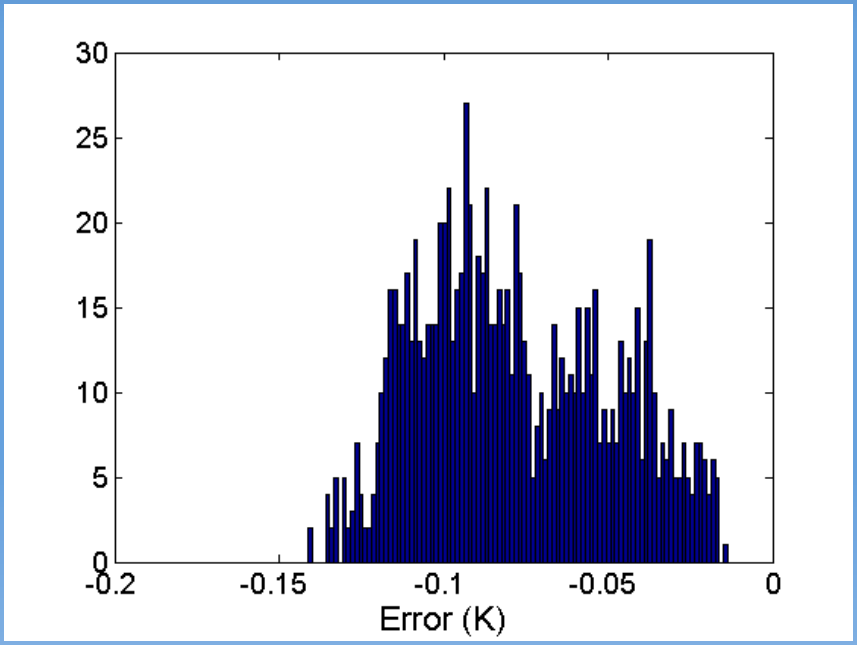
Validation I

- 32 x 32 pixel-ROI
- Fix ground temperature to 320 K
- Embed $c_{SO2} = 0.075$ at altitude of 2 km, thickness = 1 km
- Use Kilauea data set emissivity, DEM and zenith angle maps
- Calculate predicted at-sensor band radiances using MODTRAN, assuming tropical atmosphere
- Run Plume Tracker on synthetic radiance data cube

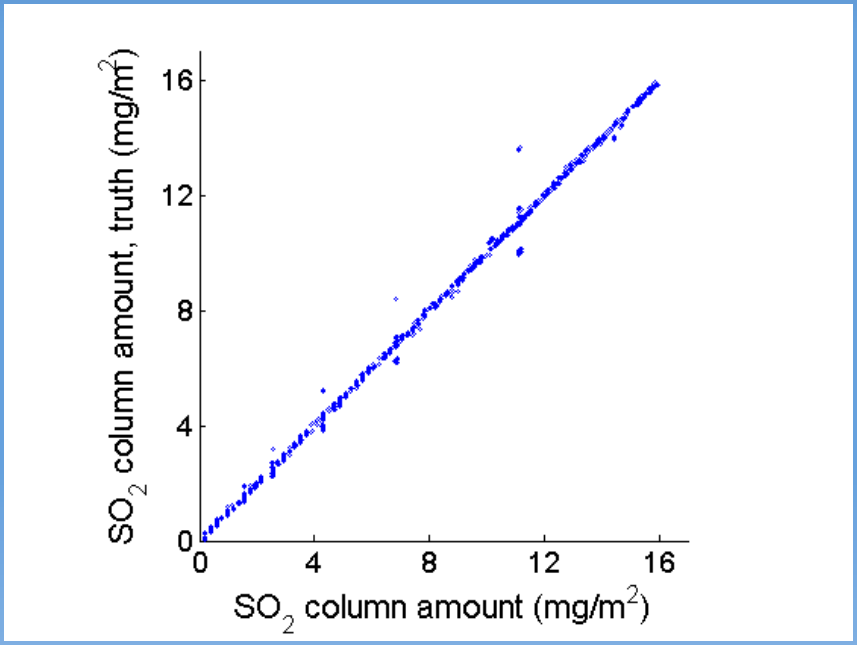


Results for validation test I

Ground temperature error distribution

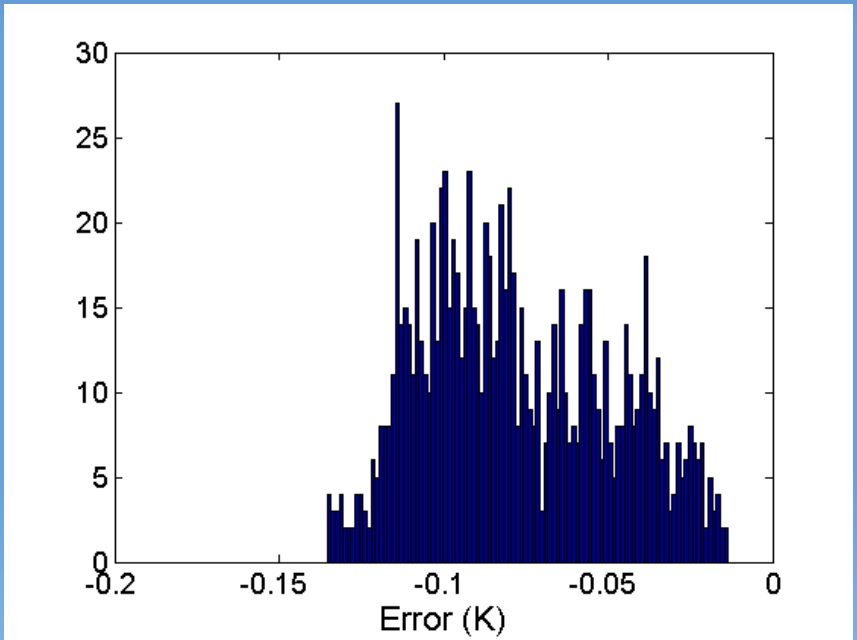


SO_2 column amount error distribution

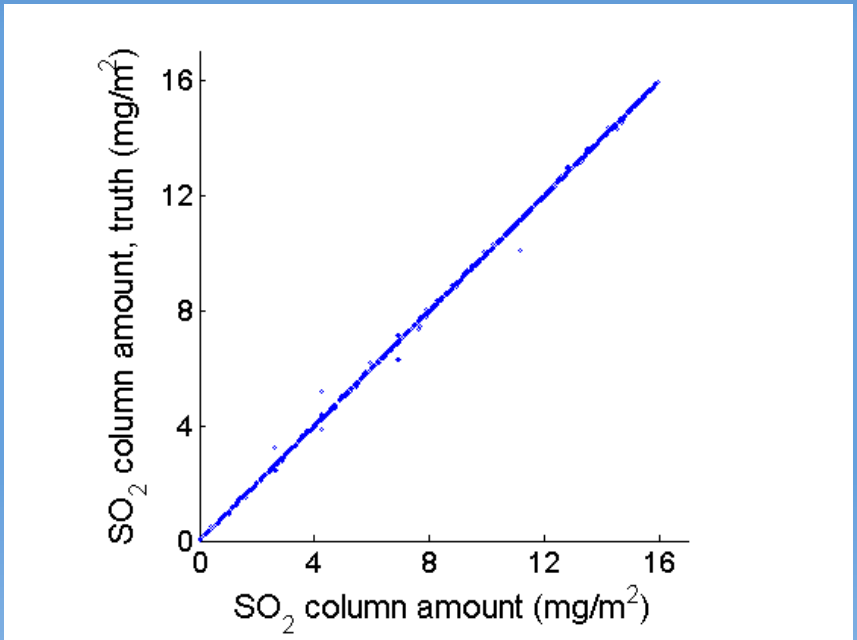


Results with finer $[SO_2]$ bins

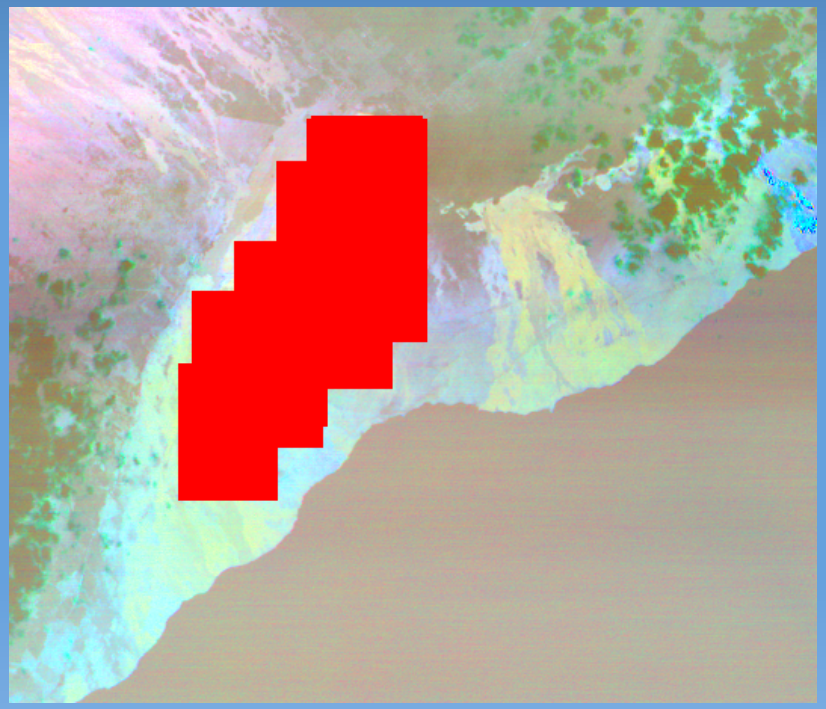
Ground temperature error distribution



SO_2 column amount error distribution

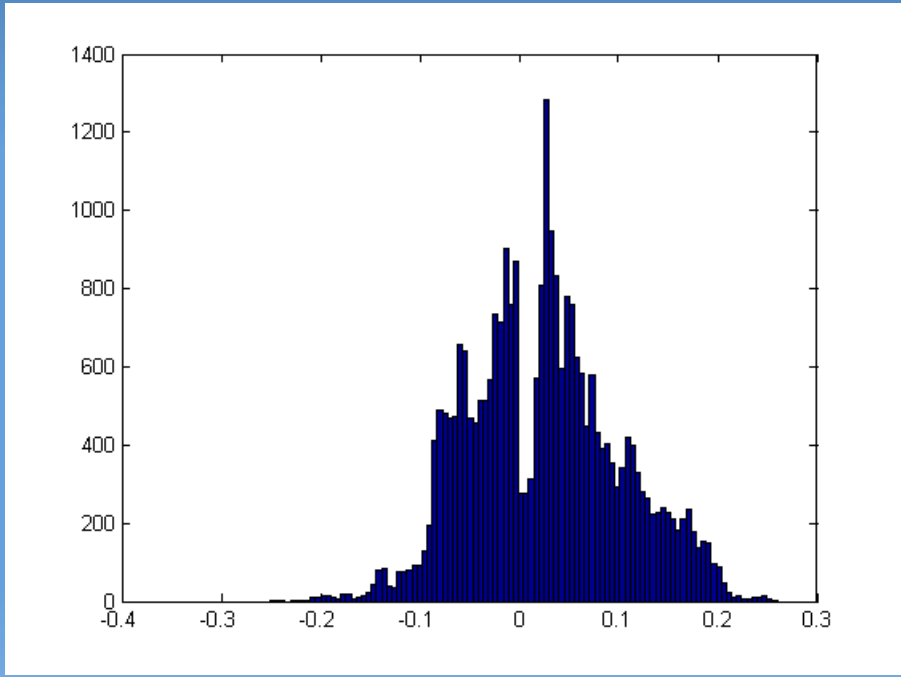


Validation II – Direct comparison with IDL-based Plume Tracker



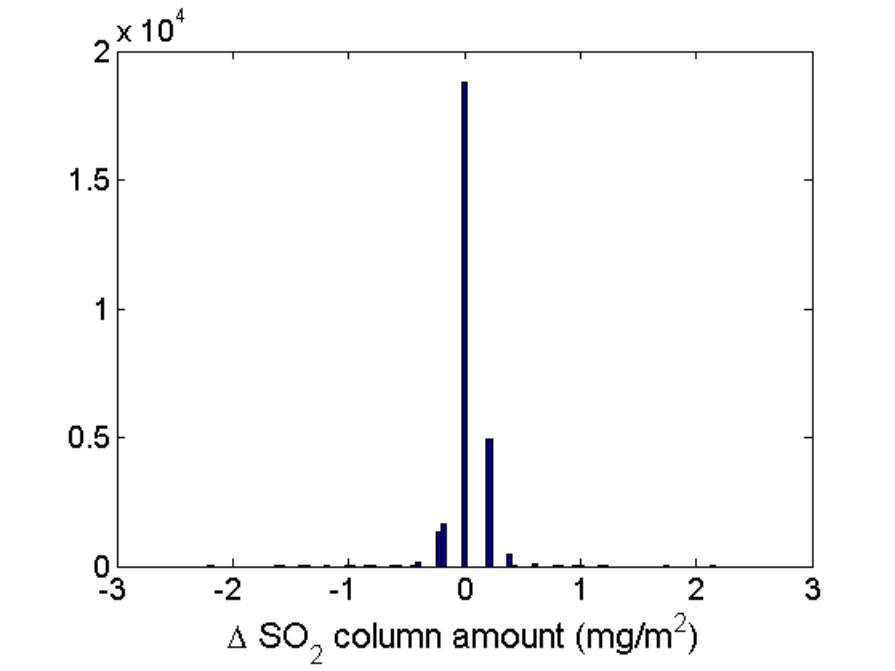
- ROI consists of 27k pixels
- T, SO_2 column amounts retrievals performed using parallel version of C++-based Plume Tracker
- ROI chosen where emissivities are defined and greater than 0.9
- MODTRAN3 version used to calculate $\overline{\mathbf{W}}$ *UD*

Validation II: Ground temperature



- Max error is at 0.26 K
- Mean error, $\langle \mathrm{abs}(DT) \rangle$ = 0.064
- New PT ground temperature estimates are on average slightly higher than those from current version

Validation II: SO_2 column amounts



- Max error of per pixel estimate is 2.0 mg/m^2
- Mean error of per-pixel estimate, $\langle abs(D SO_2) \rangle$ = 0.075 mg/m^2