Solar Event Tracking

by
Dustin Kempton
October 1, 2019
Agenda

• Motivation
• Object Tracking Background
• Input Data
• General Purpose Solar Event Tracking
• Tracking Evaluation
• Closing with Some Use Case Ideas
Why Do We Track Solar Events

We want to observe and understand solar events that have an impact on daily human activity.

Tracking is a preprocessing step for finding relations in solar event interactions.
Solar Event Tracking

Produce spatiotemporal trajectories of solar events

- Results in a dataset for pattern mining of spatiotemporal trajectories
Spatiotemporal Trajectory Sequence

- Example of simple sequence of events that can be found in data after tracking pre-processing
  - More on this later today
Object Tracking
Background
First there was Detection

• Object detection: the process of detecting, classifying, and determining the position of instances of semantic objects in digital images and videos.

• Examples of objects that are detected include:
  – Humans
  – Buildings
  – Cars
  – **Solar Events**
What Is Object Tracking

• Estimating the trajectory of an object over time by locating its position in each frame of a video sequence
What Is Object Tracking

• Estimating the trajectory of an object over time by locating its position in each frame of a video sequence
Tracking Can be used to Fix Detection Failures

• With moving objects, there are cases where visual appearance of an object is drastically different than it was in the original frame of a video

• When visual appearance differs drastically, simple detection fails, but tracking can recover from such instances
  – Motion modeling and history of an object make this possible
Where Object Tracking Can Fix Detection Failure

- Occlusion: The object in question is partially or completely occluded
- Identity switches: When two objects cross, how do you know which one is which
- Motion Blur: Object is blurred because of motion of the object or camera
- Viewpoint Variation: Different viewpoint of an object may look very different visually
- Scale Change: Large changes in object scale can cause failure to detect
- Background Clutter: Background near object has similar color or texture as the object
Underlying Modeling

• Object tracking is an old and hard computer vision problem, most techniques designed to solve it rely on two key things:
  – Motion Model: Uses past behavior of an object to predict the potential position of objects in future video frames
    • Can’t predict the abrupt change in motion and direction
  – Visual Appearance Model: Learns to discriminate the object from the background
Different Classes of Tracking Algorithms: 1

• Detection Based or Detection Free Trackers:
  – Detection Based Tracking: Video frames are given to an object detector that returns a set of detection hypotheses that are then used to form tracking trajectories.
    • In this method, the tracking algorithm is used to correct failure cases of object detection.
  – Detection Free Tracking: Requires manual initialization of a number of objects in the first frame. The objects are then located in subsequent frames by the tracking algorithm.
    • This method cannot deal with new objects appearing in the middle of a sequence of frames
Different Classes of Tracking Algorithms: 2

• Single and Multiple Object trackers:
  – Single Object Tracker: Only a single object is tracked even if the environment has multiple objects in it.
    • The object to be tracked is determined by the initialization in the first frame
  – Multi Object Tracker: All the objects present in the environment are tracked over time.
    • If a detection-based tracker is used, it can even track new objects that emerge in frames other than the first frame.
Different Classes of Tracking Algorithms: 3

• Online vs. Offline trackers:
  – Offline trackers: Used when you have to track an object in a recorded stream.
    • In this method, you can use the past frames and future frames to make more accurate tracking predictions
  – Online Trackers: Used where predictions are available immediately and hence, they can’t use future frames to improve the results.
Different Classes of Tracking Algorithms: 4

• Learning/Training Based Trackers:
  – Online Learning Trackers: Learns about the object using initialization frame and few subsequent frames. The tracker then continues to track that object using what was learned in the initialization.
  – Offline Learning Trackers: Learning is completely offline; nothing is learned at runtime of the tracking algorithm.
    • For example, we train a tracker to identify a person, then it is used to continuously track all the people in a video.
Detection and tracking of objects is an actively researched computer vision problem

• A lot of academic research focuses on human object tracking
• The main problem remains the same for all multi-object tracking
  — Data Association
Data Association Problem

• Multiple Hypothesis Data Association
  – Established track identities are assigned to detections at later frames
  – Is an unbalanced assignment problem because there may be more or fewer identities than detections in later frames
  • Unmatched tracks are pruned after some time of no further detections assigned to them
Multidimensional Assignment

• This is an extension of the multiple hypothesis assignment problem
  – Not only looking to find the most likely next detection for an existing track but the next $N$ detections
  – This is closer to our tracking algorithm
Difficulties in Tracking

• Difficulties in tracking objects can arise due to
  – Abrupt object motion
  – Changing appearance patterns of the object and/or scene the object is in
  – Non-rigid object structures
Simplifying Assumptions in Traditional Object Tracking

- Tracks are assumed to begin and end in region $E$
  - Objects do not appear or disappear in the middle of the screen
  - Objects do not split into multiple pieces
Solar Event Tracking does not have the Same Assumptions

- Solar objects can appear and disappear at, almost, any location on the screen

A. is the heat map of track starting locations
B. is the heat map of track ending locations
Additional Challenges of Tracking in Solar Domain

• Example of object splitting into two
Solar Event Tracking Input
The 2 Input Datasets

Raster Data

Vector Data
Vector = FFT Solar Events

- Majority of computer vision software/modules implemented by FFTeam do static event detection (they mark events on individual images, one at the time)

- To enable true spatiotemporal analyses of solar phenomena, we need to be able to identify phenomena as it evolves over the time
Raster = Extracted Parameters from AIA Images
Image Parameters

• Images are broken up into a grid of cells
  — Each cell is 64 pixel by 64 pixels in size
Tracking Solar Events
Iterative Refinement Tracking

• Three stages:
  • Result in iteratively growing tracks
  • Estimate from prior execution of a stage is starting point for the next
  • Start with a set of object observations from the detection modules (the dataset from HEK)
Iterative Refinement Tracking

Stage 1
- Feature Selection
  - Generate Track Fragments
  - Feature Selection from Track Fragments

Stage 2
- Track/W Visual Similarity
  - No skipped frames allowed
  - Executes only once

Stage 3
- Track/W Visual & Motion Similarity
  - Executes N Times
  - Allows Skipped Frames
Stage 1: Track Fragment Generation & Feature Selection

- Implementation for Shared Memory Parallel Random-Access Machine
  - All event detections processed in parallel
  - Allowable because of restrictions we enforce to ensure data independence
  - Duplicate results are pruned before termination
TRACK FRAGMENT GENERATION ON SHARED MEMORY PARALLEL ARCHITECTURE
Beginning State

$t_0$  $t_1$  $t_2$  $t_3$  $t_4$  $t_5$
Track Fragment Generation

- When one and only one detection falls into the search regions, we link them
  - They are also added to a track fragment
All Pairs Processed in Parallel

\[ TF_1 \]
\[ TF_2 \]
\[ TF_3 \]
Track Fragment Generation

- Multiple track fragments point to the same linked list
- We need to prune duplicate results
Track Fragment Generation

- When the fragment generation terminates, we have only one track fragment object pointing to a linked list of detections.
All Pairs Processed in Parallel
End Initial Fragment Generation Stage

Ambiguous area left out of feature selection
Feature Selection for Appearance Model

• The appearance model in tracking is used to determine the best visual match between potential matches
  – This is a classification problem

• Feature selection is used to find the most characterizing subset of image parameters for each event type
Why Feature Selection?

- There are advantages to utilizing feature selection
  - Reduction of the number of dimensions in the problem and thereby computation cost of classification
  - Reduce noise in the input data thereby increasing classification accuracy
• Given input data $D$
  – $D$ is a set of $M$ features $X = \{x_i, i = 1, \ldots, M\}$
• Given target variable $c$
  – Feature selection is finding from $\mathbb{R}^M$ a subspace of $n$ features $\mathbb{R}^n$ that “optimally” characterizes $c$
    • With $n \ll M$
Feature Generation

• Create a histogram for each event detection
  – This is done for each wavelength and image parameter pair
  – Gives 90 histograms for each event detection
Feature Generation Cont.

- Histogram similarity is measured for each pair of detections that are known to be temporal neighbors in the same tracked object
  - This gives the feature value for the same track class (SAME)
Feature Generation
Cont.

- The initial left histogram is kept, and another detection is randomly chosen for the right.
  - This gives the feature value for the different track class (DIFF)
Selection by Filtering

Features are selected using some intrinsic characteristic of the feature

- Ideally a characteristic that determines its power to discriminate between the target classes e.g., ANOVA F-test

Has the advantage of being relatively easy to compute

Uses characteristics that are uncorrelated to the learning methods that will be applied to the data
Features are ordered by their relevance to the target classes of $c$

- $F$-Statistic is used to evaluate the relevance

$$F(x_i, c) = \frac{\sum_j n_j (\bar{x}_{ij} - \bar{x}_i)^2}{(P - 1) \sigma^2}$$

- $P$ is the number of classes
- $\bar{x}_i$ is the mean value of feature $x_i$ across all classes
- $\bar{x}_{ij}$ is the mean value of feature $x_i$ within class $j$
- $n_j$ is the number of samples for class $j$
- $\sigma$ is the pooled variance across all classes
Iterative Refinement Tracking

Stage 1
Feature Selection
- Generate Track Fragments
- Feature Selection from Track Fragments

Stage 2
Track/W Visual Similarity
- No skipped frames allowed
- Executes only once

Stage 3
Track/W Visual & Motion Similarity
- Executes N Times
- Allows Skipped Frames
• Tracking using similarity
  • Input from Stage 1, has been broken back into individual detections
  • Tracking is formulated as a maximum a posteriori then solved using the minimum cost flow network algorithm
  • We currently use successive shortest paths (generalized Ford-Fulkerson algorithm)
Tracking Problem Formulation at Both Stage 2 and 3

\[ \mathcal{T}^* = \arg\max_{\mathcal{T}} P(\mathcal{T}|O) \]

- Assuming:
  - Likelihood probabilities are conditionally independent
  - Trajectories do not overlap
    \[ \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \forall k \neq l \]
  - Motion of trajectories are independent

\[ \mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(o_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k) \]
Cost-flow Network Initialization

- Two nodes $u_i, v_i$ for every track fragment $o_i \in O$ with an edge $(u_i, v_i)$
Observation Edge

- Two nodes $u_i, v_i$ for every track fragment $o_i \in O$ with an edge $(u_i, v_i)$
  - Cost to edge between them of $c(u_i, v_i) = C_i$
  - Flow $f(u_i, v_i) = f_i$
• Cost is \( C_i = \log \frac{\beta_i}{1-\beta_i} \)
  
  \( \beta_i = 1 - f(\Delta; \lambda) \) models the cases of an observation being a true detection and the cases of an observation being a false detection

  \( f(\Delta; \lambda) = \Pr(X = \Delta) = \frac{\lambda^\Delta e^{-\lambda}}{\Delta!} \), where \( \Delta \) is the absolute value of change in the number of detections from the previous frame to the frame of \( o_i \) and \( \lambda \) is the expected change

  » \( \lambda \) is calculated by looking at the previous \( n \) reporting periods from the current period.
Cost-flow Network

- There is also an edge $(S, u_i)$
  - Cost $c(S, u_i) = C_{en,i}$
  - Flow $f(S, v_i) = f_{en,i}$
Cost-flow Network

- As well as an edge \((v_i, T)\)
  - Cost \(c(v_i, T) = C_{ex,i}\)
  - Flow \(f(v_i, T) = f_{ex,i}\)
Enter/Exit models attempt to quantify the likelihood of tracks starting or ending at this report

\[ C_{en,i} = -\log P_{entr}(o_i) \]
\[ C_{ex,i} = -\log P_{exit}(o_i) \]

Both are calculated similarly

\[ P_{entr}(o_i) = \frac{L}{Gn} \]

- \( G = \text{global maximum probability} \)
- \( L = \text{local maximum probability} \)
- \( n \) be the number of pixels inside the detection
Transition Edge in a Cost-flow Network

- For every transition $P_{\text{link}}(o_j|o_i) \neq 0$, an edge is created for $(v_i, u_j)$
Transition Edge in a Cost-flow Network

- cost of $c(v_i, u_j) = C_{i,j}$
- flow $f(v_i, u_j) = f_{i,j}$
Transition Cost Function Calculations
Transition Cost

- \( P_{\text{link}}(o_j|o_i) \) changes depending upon stage
  - \( P_{\text{link},\text{stage}_2}(o_j|o_i) = P(a_j|a_i)P(\Delta t|o_i) \)
  - \( P_{\text{link},\text{stage}_3}(o_j|o_i) = P(a_j|a_i)P(\Delta t|o_i)P(v_j|v_i) \)
Models for Transition Edge Costs

• Transition model attempts to quantify the likelihood of a track continuing to the next report

\[ C_{i,j} = -\log P_{\text{link}}(o_j|o_i) \]

• Two variants of \( P_{\text{link}}(o_j|o_i) \)
  - \( P_{\text{link, stage}_2}(o_j|o_i) = P(a_j|a_i)P(\Delta t|o_i) \)
  - \( P_{\text{link, stage}_3}(o_j|o_i) = P(a_j|a_i)P(\Delta t|o_i)P(v_j|v_i) \)

• \( P(a_j|a_i) \) is the Appearance Model
• \( P(\Delta t|o_i) \) is the Frame Skip Model
• \( P(v_j|v_i) \) is the Motion Model
Appearance Modeling using Sparse Region Descriptors in Solar Event Tracking
Online Sparse Appearance Model

• We start with a solar event we wish to model.
The area of interest is represented by a matrix of cell values.

There are up to $m$ of these matrices of cell values.
Window Extraction

- A window of size $p$ by $p$ is used to construct a vector in matrix $X$
Signal Vector from Window

- Cells in the window of size $p$ by $p$ are collimated to produce vector $x_i \in X$
Rinse and Repeat

- The window is moved within the area of interest, and the construction of another vector in matrix $X$ is prepared.
Signal Matrix

- Each step of the window is another vector in matrix $X$
- For tracking $X$ is produced extracting signals using
  - overlapping windows of size 4 cells by 4 cells
  - step size when moving the window is 1 cell
Sparse Modeling

• Use extracted signal vectors to learn a dictionary and model local patches
• Once the dictionary is learned each of the signals in $X$ can be described as a linear combination of elements in $D$:

$$x_i = D\alpha_i + \epsilon$$

using

$$\min_{\alpha \in \mathbb{R}^k} \frac{1}{2}\|x - D\alpha\|_2^2 + \lambda\|\alpha\|_1$$
Short Walk Through
Train on CH

Select a window that centers on a Coronal Hole in the data
Extract Patches

• Using 3 wavelength and parameter combinations we extract patches from the window
Learn a Dictionary

Learn the dictionary of basis vectors
Map the Output-CH

- For each parameter cell on the solar disk, the probability of the learned patch being located there can be calculated.
- We limit to just the candidate regions when performed for tracking.
Histogram Comparison

- Target Model Histogram
  - Coefficient matrix \( \alpha \) of the original window is used

- Target Candidate Histogram
  - Coefficient matrix \( \hat{\alpha} \) constructed from a candidate ROI is used
  - A scaling value is also used to match the original ROI

- Use their similarity to find a likelihood value for the tracking algorithm
Target & Candidate

Target Coronal Hole

Candidate Coronal Hole

Target Weight Map

Candidate Weight Map

Target Histogram

Candidate Histogram
Target & Candidate
Cost-flow Network

• We find the greatest flow for the minimal cost.
• The track fragments that fall along each path in lowest cost set of paths, gets associated
  – Duplicates get pruned again
  – Results sent to stage three
Iterative Refinement Tracking

Stage 1
Feature Selection
- Generate Track Fragments
- Feature Selection from Track Fragments

Stage 2
Track/W Visual Similarity
- No skipped frames allowed
- Executes only once

Stage 3
Track/W Visual & Motion Similarity
- Executes N Times
- Allows Skipped Frames
The Iterative Part

1. Input Track Fragments used to Construct Graph
2. Transition Edges are Added for Candidates
3. Increase Allowed Skipped Frames
4. Cost Flow Problem Solved & Fragments in Each Path are Linked

Repeat Until Skip = N
Tracking Evaluation
Human Labeling for Validation

- Used non-expert human labelers
  - Labelers select from a set of potential paths
  - Use a quorum voting to determine path of trajectories
Human Labeling Example

- Target region and search area top
- Two candidate regions bottom
Human Labeler Disagreement

Active Regions

Coronal Holes

Labelers tend to disagree at the limbs
Labeler Coverage

- Minimum user input per event report is 3
  - Allows for majority win voting (unless even votes are split)
Percentages from Label Results

• Users agree on label for next event a majority of the time
  – Over 50% of the CH dataset has 100% user agreement
  – About 40% of the AR dataset has 100% user agreement
Confidence from Label Results

• Even when there is disagreement one report usually stands out
  – Almost all reports have more than 50% of votes going to a single next report
## Tracking vs. SPoCa

### Tracking Results using Human Labeled Ground Truth

<table>
<thead>
<tr>
<th>Data</th>
<th>MOTA</th>
<th>MT%</th>
<th>Switches</th>
<th>Track Count</th>
<th>Avg. Length</th>
<th>Max Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPoCA 100% RSUN</td>
<td>0.902</td>
<td>0.662</td>
<td>0.681</td>
<td>485</td>
<td>5.45</td>
<td>73</td>
</tr>
<tr>
<td>80% RSUN</td>
<td>0.876</td>
<td>0.643</td>
<td>0.736</td>
<td>323</td>
<td>4.18</td>
<td>44</td>
</tr>
<tr>
<td>Track 100% RSUN</td>
<td>0.923</td>
<td>0.675</td>
<td>0.541</td>
<td>399</td>
<td>6.63</td>
<td>75</td>
</tr>
<tr>
<td>80% RSUN</td>
<td>0.925</td>
<td>0.736</td>
<td>0.445</td>
<td>221</td>
<td>6.11</td>
<td>44</td>
</tr>
<tr>
<td>CH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPoCA 100% RSUN</td>
<td>0.976</td>
<td>0.889</td>
<td>0.156</td>
<td>130</td>
<td>12.01</td>
<td>108</td>
</tr>
<tr>
<td>80% RSUN</td>
<td>0.986</td>
<td>0.842</td>
<td>0.193</td>
<td>58</td>
<td>13.88</td>
<td>46</td>
</tr>
<tr>
<td>Track 100% RSUN</td>
<td>0.969</td>
<td>0.877</td>
<td>0.197</td>
<td>220</td>
<td>7.1</td>
<td>108</td>
</tr>
<tr>
<td>80% RSUN</td>
<td>0.991</td>
<td>0.895</td>
<td>0.123</td>
<td>49</td>
<td>16.43</td>
<td>46</td>
</tr>
</tbody>
</table>

SPoCa [Verbeeck et al. 2014]
And this is our After

Active Regions in Jan, 2012

Coronal Holes in Jan, 2012
Some Tracking Use Cases
A Tracked Object
Use: Path Mapping

• Map paths of different subsets of Solar events.
  – Flaring Active Region (Top)
  – Non-Flaring Active Regions (Bottom)
Tracked Object Use: Time Series Analysis

- Time Series Analysis of the visual changes in evolving regions.
  - How does an active region change visually prior to some interesting event?
Visualizations

FOLLOWING VISUALIZATIONS
USE 6 MONTHS OF DATA

Kempton
AR Heat Map

Start of tracks
AR Heat Map

- End of tracks.
CH Heat Map

• Start tracks.
• End tracks.
• Start tracks.

SG Heat Map
• End tracks.

SG Heat Map
SS Heat Map

• Start tracks.
SS Heat Map

- End tracks.
All Heat Map
AR Center X value vs. Time
AR Center Y value vs. Time
CH Center Y value vs. Time
SS Center X
value vs. Time
SG Center Y
value vs. Time
Data Access Optimizations

• We can exploit the fact that tracks from one month don’t interact with tracks from another month
  – Process one month at a time
  – Cache images while processing one month (Order of magnitude faster than disk access for each iteration)
  – Old images are purged from LRU cache while loading image for the next month for the first time.
Software Developed For this Work

- **DMLabLib** – Is a cross project library of our most used code. Link: [https://bitbucket.org/gsudmlab/dmlablib](https://bitbucket.org/gsudmlab/dmlablib)

- **SolarTracking** – The original release of the offline tracking algorithm. (Outdated) Link: [https://bitbucket.org/gsudmlab/solartracking](https://bitbucket.org/gsudmlab/solartracking)

- **Part of ISD Processing Pipeline** *(More about this tomorrow)*
  
  - **HEK-Processor** – Program to periodically pull metadata from the HEK and store it locally. Link: [https://bitbucket.org/gsudmlab/hek-processor](https://bitbucket.org/gsudmlab/hek-processor)
  
  - **Image-Processor-Server** – Part one (server) of a server worker pair used to distribute and process images to obtain our image parameter dataset. Link: [https://bitbucket.org/gsudmlab/image-processor-server](https://bitbucket.org/gsudmlab/image-processor-server)
  
  - **Image-Processor-Worker** – Part two (worker) of the server worker pair used to distribute and process images to obtain our image parameter dataset. Link: [https://bitbucket.org/gsudmlab/image-processor-worker](https://bitbucket.org/gsudmlab/image-processor-worker)
  
  - **Tracking-Processor** – Updated tracking algorithm to handle both historic metadata and the current metadata, to update the tracking dataset as new data becomes available. Link: [https://bitbucket.org/gsudmlab/tracking-processor](https://bitbucket.org/gsudmlab/tracking-processor)
Publications From This Work


Any Questions?

Thank you for your attention.
Dustin Kempton, Ph.D.
Department of Computer Science
Georgia State University
dkempton1@cs.gsu.edu