8-7-2018

Sparse Coding for Event Tracking and Image Retrieval

Dustin J. Kempton
Georgia State University

SPARSE CODING FOR EVENT TRACKING AND IMAGE RETRIEVAL

by

DUSTIN KEMPTON

Under the Direction of Rafal Angryk, PhD

ABSTRACT

Comparing regions of images is a fundamental task in both similarity based object tracking as well as retrieval of images from image datasets, where an exemplar image is used as the query. In this thesis, we focus on the task of creating a method of comparison for images produced by NASA's Solar Dynamic Observatory mission. This mission has been in operation for several years and produces almost 700 Gigabytes of data per day from the Atmospheric Imaging Assembly instrument alone. This has created a massive repository of high-quality solar images to analyze and categorize. To this end, we are concerned with the creation of image region descriptors that are selective enough to differentiate between highly similar images yet compact enough to be compared in an efficient manner, while also being indexable with current indexing technology. We produce such descriptors by pooling sparse coding vectors produced by spanning learned basis dictionaries. Various pooled vectors are used to describe regions of images in event tracking, entire image descriptors for image comparison in content based image retrieval, and as region descriptors to be used in a content based image retrieval system on the SDO AIA image pipeline.

SPARSE CODING FOR EVENT TRACKING AND IMAGE RETRIEVAL

by

DUSTIN KEMPTON

A Dissertation Submitted in Partial Fulfillment for the Degree of

Doctor of Philisopy

in the College of Arts and Sciences

Georgia State University

2018

SPARSE CODING FOR EVENT TRACKING AND IMAGE RETRIEVAL

by

DUSTIN KEMPTON

Committee Chair: Rafal Angryk

Committee: Rajshekhar Sunderraman
Zhipeng Cai
Petrus Martens

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
August 2018

DEDICATION

This work is dedicated to the most supportive and understanding person I know, Rachel. Her seemingly unending patience has been invaluable to me as I have worked towards this goal. She has helped me stay the course, and for that I will be ever grateful.

ACKNOWLEDGEMENTS

The work presented in this dissertation was made possible with the support of many people. First, I wish to express sincere gratitude to my adviser Dr. Rafal Angryk; it was his trip to the University of South Dakota to recruit students that motivated me to pursue this course of study under his tutelage. Dr. Angryk's confidence in my abilities and a small bit of helpful nagging about consistently publishing have helped me get to this point. Thank you.

I would also like to thank each of my committee members – Dr. Rajshekhar Sunderraman, Dr. Piet Martens, and Dr. Zhipeng Cai, for their continual support.

I appreciate all the discussions, feedback, assistance, and friendship from the members of our lab over the years–Berkay, Mike, Azim, Soukaina, Sushant, Ruizhe, Ahmet, Hamdi, Vijay, Karthik, Thaddeous, Sunitha, Sajitha, and Soumitra.

I would be remiss if I didn't mention Dr. Doug Goodman and Dr. Jose Flores from my undergraduate studies at the University of South Dakota. Their passion for Computer Science and Mathematics inspired my interest in these fields and were instrumental in my desire to persist in my studies.

FUNDING ACKNOWLEDGEMENT

Work for this dissertation has been supported in part by funding from the Division of Advanced Cyberinfrastructure within the Directorate for Computer and Information Science and Engineering, the Division of Astronomical Sciences within the Directorate for Mathematical and Physical Sciences, and the Division of Atmospheric and Geospace Sciences within the Directorate for Geosciences, under NSF award #1443061. It was also supported in part by funding from the Heliophysics Living With a Star Science Program, under NASA award #NNX15AF39G.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Comparing regions of images is a fundamental task in both similarity based object tracking and retrieval of images from image datasets by using an exemplar as the query, commonly called Content Based Image Retrieval. The process of tracking multiple objects in video data can be described as a task of linking detections over time into the most likely trajectories, based on a model of likelihood. The likelihood calculation generally includes the prior probability of the presence of the target (object being tracked), and how likely the detection is to be a false detection. Additionally, the likelihood calculation generally includes several methods of comparing similarity between the potential detections in a particular path [1], including visual similarity. In Content Based Image Retrieval, a task that is frequently performed is the k-nearest neighbor (kNN) search, in which the database is queried for the k most similar records to any given query point. Indexing the data is often used for efficient retrieval in such queries. However, most indexing techniques degrade quickly as the number of dimensions increase [2], and even state of the art high-dimensional indexing techniques perform no better than performing a sequential scan beyond 256 dimensions [3].

The focus of this thesis is to create methods of comparing image regions from instruments on NASA's Solar Dynamic Observatory (SDO) mission. We will focus on one of the crucial remaining steps needed to create an image retrieval system for this large, and growing, dataset. Specifically, we are concerned with the creation of image region descriptors that are both selective enough to differentiate between highly similar images, and compact enough to be indexed using current indexing techniques. As the ground work for these region descriptors, we present our current contributions of utilizing sparse coding for creating descriptors for regions of images used in event tracking,

and for the creation of a selective entire-image descriptor used to compare entire images for content based image retrieval. These use cases are steps towards our final goal of building and analyzing region descriptors for a content based image retrieval system on the SDO AIA image pipeline.

1.1 Motivation

In February 2010, NASA launched the Solar Dynamics Observatory, the first mission of NASA's Living with a Star program, which is a long term project dedicated to the study of the Sun and its impacts on human life [4]. The SDO mission is an invaluable instrument for researching solar activity which can produce damaging space weather. This space weather activity can have drastic impacts on space and air travel, power grids, GPS, and communications satellites [5]. For example, in March of 1989, geomagnetically induced currents, produced when charged particles from a coronal mass ejection impacted the earth's atmosphere, caused power blackouts and direct costs of tens of millions of dollars to the electric utility "Hydro-Qubec", due to equipment damage and lost sales [6]. If a similar event would have happened during the summer months, it is estimated that it would likely have produced widespread blackouts in the northeastern United States, causing an economic impact in the billions of dollars [6].

The SDO represents a new frontier in quantity and quality of solar data, with approximately 70,000 full disk, high-resolution (4096 by 4096 pixel) images of the Sun per day. In total, the SDO produces about 1.5 TB per day, or about 0.55 PB of data per year, and has already produced more data than all previous solar data archives combined [7, 8]. This data is collected by three independent instruments on the satellite: the Extreme Ultraviolet Variable Experiment (EVE), which takes measurements of the spectral distribution of extreme ultraviolet radiation propagating from the sun [9], the Helioseismic and Magnetic Imager (HMI), which captures the motion of the sun's surface and measures the surface magnetic field [10], and the Atmospheric Imaging Assembly (AIA), which cap-

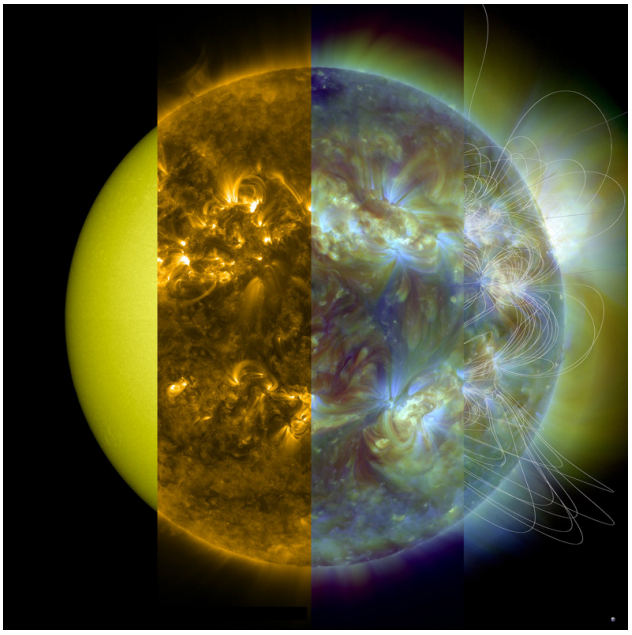


Figure 1.1: Visualization of the layers of the Sun through various SDO data and metadata. Courtesy of NASA/SDO and the AIA, EVE, and HMI science teams.

tures full-disk images of the sun in ten separate electromagnetic wavelength bands across the visual and ultra-violet spectra, each selected to highlight specific elements of known solar activity [11]. An example of how these various wavelengths represent different layers of the Sun is shown in Figure 1.1, which also includes extrapolated magnetic field lines on the rightmost slice.

There have been previous explorations of creating content based image retrieval systems for this dataset, which focused on retrieval of similar entire images [12]. There were later updates that started to explore region retrieval [13, 14]. However, the retrieval was limited to only comparing labeled regions, and was implemented with lookup tables of precomputed nearest neighbors. This approach did not lend itself to use on constantly growing datasets, nor for the querying of regions that were outside of the labeled subset. These previous works utilized 10 different texture-based image parameters that have been calculated and stored for images at a 6 minute cadence. These parameters were chosen because of their ability to be quickly calculated and their demonstrated usefulness on images of similar structure, such as medical X-rays [15]. These image parameters are

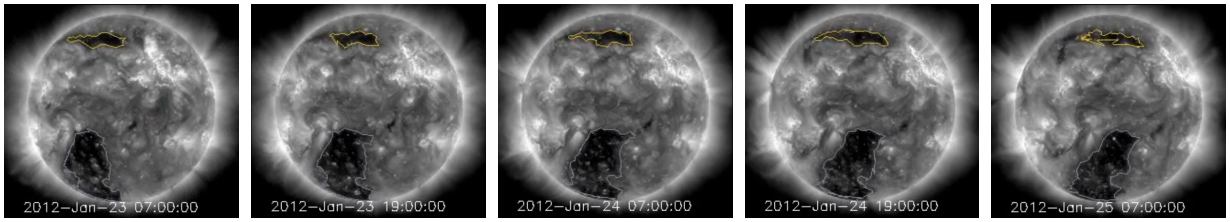


Figure 1.2: Sequence of two tracked Coronal Holes over five reporting times.

utilized, instead of the original images, because the storage of the original image data has rapidly become both computationally prohibitive and costly for on site processing. This is due to the fact that the AIA instrument surpassed its 100 millionth image on January 19, 2015, marking an excess of a petabyte of data. Therefore, the use of these image parameters, in lieu of the original images, saves on storage costs and allows for quicker processing when making visual comparisons. Even so, as mentioned above, these parameters did not go far enough in creating methods and tools to thoroughly and easily explore this stream, and therefore the potential for insight discovery from this data has seen limited exploration.

The purpose of the work presented in this document is to develop an image descriptor for the SDO's AIA images, through the use of sparse coding. The descriptors will have a constraint on their dimensionality so as to allow for indexing of the image descriptors for both whole images and for sub-regions of images. Such descriptors are vital for efficient searches in large collections of images, such as that produced by the SDO AIA instrument.

1.1.1 Solar Event Tracking Application for Sparse Coding

While the primary application context for sparse coded regional image descriptors will be for efficiently indexing image regions for comparison in content based image retrieval (CBIR), they can also be applied to different domains. For example, tracking of solar events is accomplished by providing a comparison mechanism to differentiate between

object detections in later images and determine which is the most likely path of an object in a previous image (See Figure 1.2). In [16], we presented a method based on sparse coding that was capable of differentiating between different classes of the same type of solar event when comparing SDO AIA images. The classes were constructed as 1) being two sequential events from the same track, and 2) being one of the two sequential events and a second event being from a different track. This construction was designed to show the applicability of the sparse coding method to the solar event tracking problem.

This sparse coding method is an improvement over the tracking work we previously produced in [17] and [18] for two reasons. The first is that it is more selective than comparing histograms of image parameters, as was done in [17] and [18]. The second is that the method of comparison we presented in [16] is an online method, requiring no previous tracked information. Whereas, the method used in [17] and [18] is an offline method, which needs previous tracking ground truth information to learn distributions of histogram distances.

1.2 Challenges

The task of describing image regions for indexing and retrieval in a CBIR system is difficult, as it requires the characterization of images with efficiently comparable vectors. In general, CBIR refers to feature recognition and image categorization using the image data alone, and does not have supporting metadata such as descriptions, captions, legends, *etc*. The creation of the feature vectors occur at the intersection between two opposing goals of: (1) accurately describing an image with increasing numbers of features to provide detailed information about the content of the image. (2) limiting the number of features to allow for efficient comparison and indexing. In the first goal, we wish to describe an image region accurately enough to be able to discern subtle differences between the millions of regions of interest in our dataset. However, increasing the

number of visual features in a single feature vector severely reduces the performance and effectiveness of the distance measure method [19].

1.3 Outline

The rest of this document is organized in the following manner. In Chapter 2, the background information on the input image parameter data, sparse coding, image retrieval, and solar event tracking is presented. This will include the definitions for basis vector dictionaries, signal vectors, and other relevant information related to sparse coding, image retrieval, and solar event tracking. Then, in Chapter 3, the foundational feature selection methods used for selecting a subset of image parameters in solar event region comparison is presented. The feature selection methods are then incorporated into the tracking methods presented in Chapter 4, with the goal of the solar event tracking having the ability to determine the most likely path a tracked solar event takes in later time steps of an input image sequence.

After tracking is introduced as utilizing the sparse coding methods for region comparison, we show how the feature selection methods are used in our work on describing solar images for similarity search in Chapter 5. In describing solar images, we begin with whole image comparison in Section 5.1, with the goal of comparing images of a single wavelength. In Section 5.2, we focus on region comparison, which has the goal of comparing similar regions without having the constraint of using a single wavelength, similar to the goal in tracking. Then, in Chapter 6, we present several evaluations of our methods, beginning with the evaluation of using these sparse coding methods in tracking in Section 6.1. In Section 6.2, we present the evaluation of our whole image comparison methods, followed by the evaluation of our region comparison method evaluations in Section 6.3. Finally, in Chapter 7, we conclude this thesis with a summary of our findings, and discuss future work.

2 BACKGROUND

This chapter is intended to present information about the input data we utilize as well as to serve as an introduction to some of the data processing methods that are used in this dissertation. It begins by providing some details about the input image data in Section 2.1, which is a set of texture-based image parameters for SDO AIA images. Then, in Section 2.2, the process of sparse coding that will be used throughout the remainder of this dissertation is described. This includes the description of an extraction process, in Section 2.2.1, which is applied on the image parameter data in areas of interest so that they can be used as input in the process of dictionary learning. Then, in Section 2.2.2, a brief discussion of the Lasso problem is given. This problem is central to both the dictionary learning processes used to create a set of basis vectors, which is discussed in Section 2.2.3, and the subsequent sparse coding of input signals over the learned dictionary. Finally, this chapter is closed with brief introductions of two application areas in which we utilize these sparse coding methods. The first is tracking solar events in Section 2.3, and the second is the comparison of images for image retrieval in Section 2.4. Though we introduce these two subjects in this chapter, they are covered in more detail in later chapters.

2.1 Image Parameter Data

As was mentioned in Section 1.1, in preparing for a CBIR system on the images produced by the AIA instrument, [22–24] found that texture-based image parameters work well for image comparison and were later shown to be useful in identifying similar regions of AIA images in [13] and [20]. The parameters that were chosen are Entropy, Mean, Standard Deviation,Fractal Dimension, 3^{rd} Moment (skewness), 4^{th} Moment (kur-

Table 2.1: Image parameters produced by [22], where M stands for the number of pixels in an image cell, and L is the maximum intensity value possible for images in the dataset. z_j is the intensity value of the j^{th} pixel, and $p(i)$ is the probability (or normalized frequency) of intensity value $i \in [0, 255]$ for the set of z_j in the cell. The fractal dimension is calculated on the box-counting method where $N(\varepsilon)$ is the number of boxes of side length ε required to cover the image cell.

Name	Equation
Entropy	$-\sum_{i=0}^{L} p(i) \cdot \log_2(p(i))$
Mean (μ)	$\sum_{i=0}^{L} p(i) \cdot i$
Standard Deviation (σ)	$\sqrt{\frac{1}{M}\sum_{j=0}^{M-1}(z_j - \mu)^2}$
Fractal Dimension	$-\lim_{\varepsilon \to 0}(\frac{\log N(\varepsilon)}{\log(\varepsilon)})$
Skewness (μ_3)	$\frac{1}{\sigma^3}\sum_{j=0}^{M-1}(z_j - \mu)^3 p(z_j)$
Kurtosis (μ_4)	$\frac{1}{\sigma^4}\sum_{j=0}^{M-1}(z_j - \mu)^4 p(z_j)$
Uniformity	$\sum_{j=0}^{M-1} p^2(z_j)$
Relative Smoothness	$1 - \frac{1}{1+\sigma^2}$
Tamura Contrast	$\frac{\sigma^2}{\mu_4{}^{0.25}}$
Tamura Directionality	see [20, 21]

tosis), Uniformity, Relative Smoothness, Tamura Contrast, and Tamura Directionality. The formulas for the selected parameters are listed in Table 2.1.

The dataset created by the previous list of parameters is created from images captured by the SDO spacecraft, and are extracted from the AIA images at a six-minute cadence for each wavelength. The original images are high resolution (4096×4096 pixel), full-

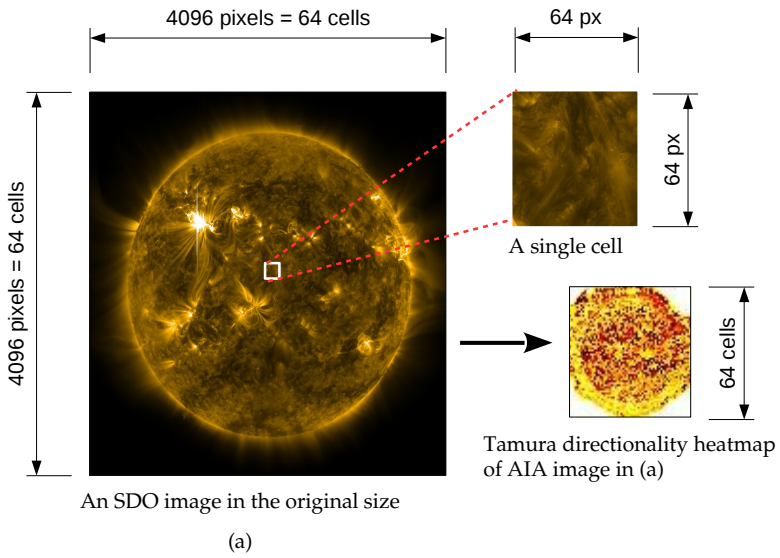


Figure 2.1: An illustration that shows the size of the original images, one cell as a sample, and the final result of calculating Tamura directionality over each and every cell. Every pixel in the heatmap represents the directionality degree calculated for the corresponding segment in the original image.

disk snapshots of the Sun, taken in ten extreme ultraviolet channels (the nine channels that we utilize in this work are 94Å, 131Å, 171Å, 193Å, 211Å, 304Å, 335Å, 1600Å, and 1700Å) [11]. These original high resolution images are accessible upon request from the Joint Science Operations Center[1]. Interested parties can also utilize a random access API for smaller files that have been compressed into high resolution JP2 representations of the original data at the Helioviewer repository[2]

The first step in calculating the image parameters involves extracting a 64 by 64 pixel cell, as shown in Figure 2.1. This is repeated to create a total of 4096 non-overlapping cells, as can be seen in Figure 2.2. Then, each of the 10 parameters is calculated using only the pixel values within the respective cell that the value is going to represent. This segmentation process was shown to be an effective approach in [22] and [25] and reduces the full-resolution images to 10 sets of a 64 by 64 cell grid of parameter values for each image coming from the SDO AIA instrument. In Figure 2.3, we show how these 10

1 http://jsoc.stanford.edu
2 https://api.helioviewer.org

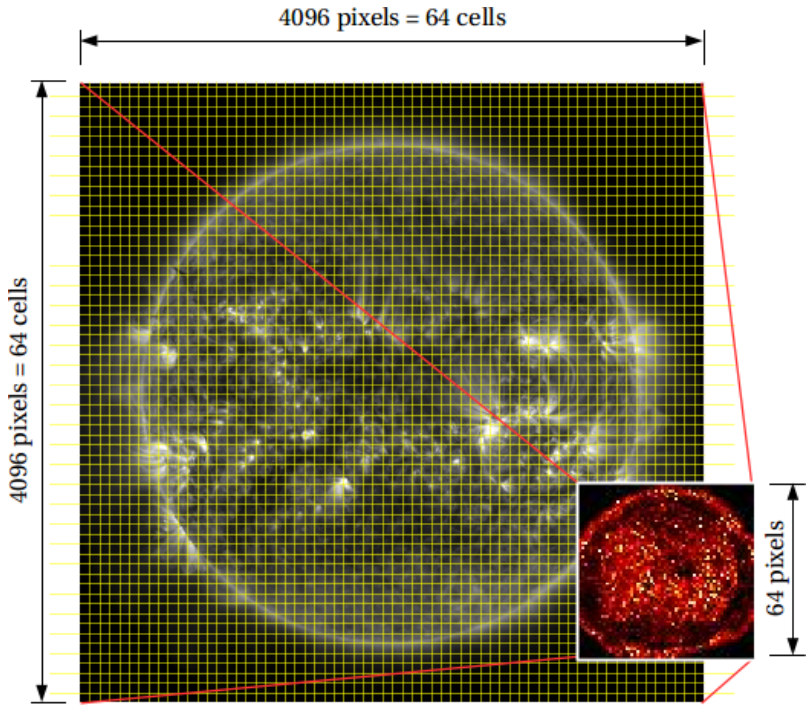


Figure 2.2: The segmentation of an original-sized sample AIA image, and the visualization of Tamura directionality where every pixel represents the degree of directionality of the corresponding cell in the original image.

sets of parameters stack to produce an image data cube for each image. A set of these parameter values was calculated over a subset of the images captured by SDO and made available by [26]. Later, this set was extended to encompass a larger date range in [27–29], and finally made available online through an API[3] by [30].

2.2 Sparse Coding

Sparse coding is a term used to identify a class of unsupervised methods that are used for learning dictionaries of basis vectors in order to represent data efficiently. In these methods, the input signals are decomposed into linear combinations of a few basis vectors from the learned dictionaries. Sparse coding is unlike decompositions based on principal component analysis in that the basis vectors need not be orthogonal, which allows more flexibility to adapt the representation to the data. Also, the elements of the

3 http://api.isd.dmlab.cs.gsu.edu/

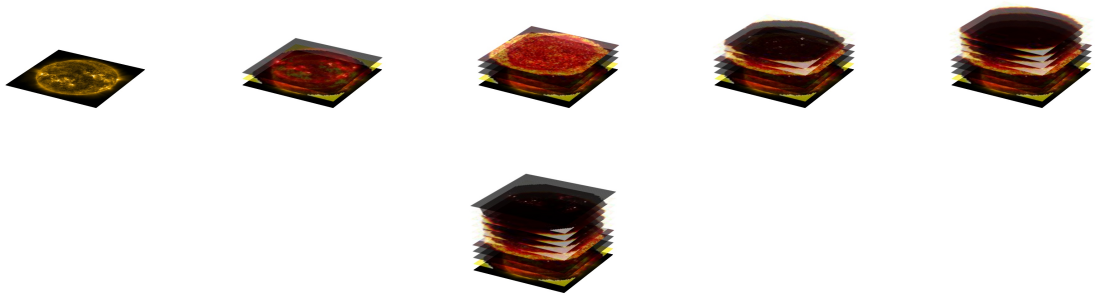


Figure 2.3: 10 sets of image parameters stacked to produce a cube of image data for an image of the 171Å filter waveband.

dictionary are not necessarily linearly independent and the set of basis vectors in the dictionary are allowed to have more elements than the signal dimension [31].

One of the main sub-problems of sparse coding is that of sparse representation, where it is desired to infer an unobserved high-dimensional state of the world from a limited number of observations. In this inference problem, there is an underlying assumption of simplicity, in that it is assumed that a signal vector $x \in \Re^m$ can be represented as the linear combination of a small number of basis vectors. For example, it is hoped that not all of the 30,000 or so genes in the human body are directly involved in the process that leads to the development of cancer, or utilizing customer ratings on perhaps 50 to 100 movies would be sufficient to predict other preferred movies [32].

In the sparse representation sub-problem for sparse coding, the decomposition is over a basis dictionary or codebook set with k entries $D \in \Re^{m \times k}$ with some noise ϵ:

$$x = D\alpha + \epsilon \tag{2.1}$$

In the decomposition, the coefficients of α for the given input vector x are assumed to have only a few coefficient weights that deviate from zero in any significant way, or are sparse. In other words, it is assumed that only a small subset of variables is truly important in a specific context. To accomplish this sparsity constraint, an optimization

problem with ℓ_1 regularization is utilized that penalizes the coefficients of α for being far from zero, since it is well known that this ℓ_1 regularization yields a sparse solution for α. By utilizing this regularization problem, the linear model $x \approx D\alpha$ is transformed to the quadratic program:

$$\underset{\alpha \in \Re^k}{\operatorname{argmin}} \frac{1}{2}\|x - D\alpha\|_2^2 + \lambda\|\alpha\|_1 \tag{2.2}$$

where λ is a regularization parameter. The sparse coding problem in Equation 2.2 is a constrained version of the ordinary least squares regression problem known as basis pursuit [33], or the Lasso [34]. This ℓ_1 regularization problem is generally used because it yields a sparse solution for α. We find the optimal values of the coefficients of α in Equation 2.2 by using the Least Angle Regression (LARS) algorithm of [35]. By using the Lasso, we search for the "best matching" projection of the multi-dimensional input vector x onto our dictionary D, while keeping the solution sparse.

In the following subsections, we will give some background information about our process of producing sparse coding vectors. We begin by discussing the process of extracting a signal vector x from an area of interest in Section 2.2.1. The discussion of the extraction process is followed up by an explanation of the Lasso problem in Section 2.2.2, as it is used for dictionary learning and the final representation of the extracted signals over the learned dictionary. Then, we discuss the process of learning the dictionary from a set of input signal vectors in Section 2.2.3. Note that the set of input signal vectors $\mathcal{X}$ is produced in different ways dependent upon the application area. However, for the current discussion, we can ignore the exact content of the signal matrix and discuss its construction with more detail in the appropriate sections dealing with the specific application.

2.2.1 *Image Parameter Layer Extraction*

The first step necessary in the construction of sparse coded vectors over a basis dictionary is to extract a set of signal vectors from the source data. Though the exact construction of the input matrix differs slightly from one application to the next, each follows the general methods set forth in this section. As was mentioned in Section 2.1, the original images from the SDO AIA instrument are 4096 by 4096 pixel images which are transformed to 10 sets of texture parameters in 64 by 64 pixel cells, listed in Table 2.1. For the current discussion, however, it is sufficient to say that a single signal vector $x \in \Re^m$, is constructed by utilizing a sliding window of size p by p. This is done by placing a sliding window over a portion of the area of interest, and then copying the cell values within the window to a column vector, with each feature value inside the window concatenated onto the end of the column vector produced by the previous extracted value. For example, using one wavelength of the available images and all 10 parameters from that wavelength, a patch sized 4 by 4 cells would result in 16 cell values from the first parameter, followed by 16 from the second, and so on until all 160 values have been copied into the signal vector, denoted as x. For this example, the length of the signal $x \in \Re^m$ is calculated as $m = p \times p \times 10 \times (number\ of\ wavelengths\ used)$ or $m = 4 \times 4 \times 10 \times 1 = 160$.

In order to construct the next entry in the set of input signals, the extraction process is repeated after sliding the patch by a predetermined number of cells within the area of interest and extracting the next signal. The set of all signals extracted from the sliding and extracting process is then denoted as $\mathcal{X} = \{x_i | i = 1 : n\}$, where n is the number of patches that were extracted by the process. It is this set of extracted column vectors that is used in the basis dictionary learning process. The learning of the basis dictionary $D \in \Re^{m \times k}$, where k is the chosen number of basis vectors to learn, is described below in Section 2.2.3.

2.2.2 *Lasso*

Much of the work of learning dictionaries and using them to produce a sparse representation vector relies on a linear model used for regression analysis called the least absolute shrinkage and selection operator (Lasso). In a typical linear regression model, it is assumed that there are N observations of an outcome (target and dependent) variable x_i, and k associated predictor (explanatory and independent) variables (or features) $d_i = (d_{i1}, \ldots, d_{ik})^T$. The goal of the model is to predict the outcome from the predictors, for both the actual data to find which predictors play an important role and for future data to predict future outcomes [32]. With a linear regression model, the assumption is that

$$x_i = \alpha_0 + \sum_{j=1}^{k} x_{ij}\alpha_j + \epsilon_i \tag{2.3}$$

where α_0 and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_k)$ are unknown parameters and ϵ_i is an error term. The popular least squares estimate method produces parameters by minimizing the least squares objective function:

$$\underset{\alpha_0, \alpha \in \Re^k}{\operatorname{argmin}} \sum_{i=1}^{N} (x_i - \alpha_0 - \sum_{j=1}^{k} d_{ij}\alpha_{ij})^2 \tag{2.4}$$

This typically leads to all of the parameter estimates being a non-zero value, which makes the interpretation of the final model challenging when k is large. Additionally, if $k > N$, the least-square estimates are not unique, leading to an infinite set of solutions that make the objective function equal to zero, which leads to over-fitting the data [32]. In order to mitigate the over fitting and interpretability problem, a constraint is placed on the least-square solution. In the Lasso or ℓ_1-regularized regression, the ℓ_1 norm of α is limited to a user defined value, giving us the quadratic program problem in Equation 2.2.

For the problems that are of concern in sparse representation over a learned dictionary, the value of x_i is in the signal vector $x \in \Re^m$ and d_i is a row of the dictionary $D \in \Re^{m \times k}$, which we will learn. The predictors in D are considered to be standardized so that each column is centered around zero ($\frac{1}{m}\sum_{i=1}^{m} d_{ij} = 0$) and has unit variance ($\frac{1}{m}\sum_{i=1}^{m} d_{ij}^2 = 1$). Also, the outcome values x_i have been centered, meaning that ($\frac{1}{m}\sum_{i=1}^{m} x_i = 0$). This is done so that the intercept term α_0 in the optimization can be omitted. Once these pre-conditions are met, we use the Least Angle Regression (LARS) method for solving the Lasso with squared-error loss, also known as the homotopy approach.

The LARS approach is presented in Algorithm 2.1. It is related to the classic model-selection method known as Forward Selection, also called "Forward Stepwise Regression" [35]. The algorithm starts with selecting predictor d_j from the set of possible predictors D, which has the largest absolute correlation with the response vector x. With the predictor selected, the algorithm then performs a simple linear regression of x along the direction of d_j. Once this is completed, a residual vector that is orthogonal to d_j is remaining, which is called the response. The remaining predictors are then projected orthogonally to d_j and the selection process is repeated. This method leads to a set of k predictors $d_1, d_2, \ldots, d_k$ after k steps, which are then used to construct a k-parameter linear model [35].

As described so far, the Forward Selection process of LARS can be overly greedy in its fitting. So, the LARS algorithm limits its step size along predictor d_j. This is similar to Forward Stagewise Selection in that Stagewise also limits the step size and proceeds by taking thousands of tiny steps as it moves toward a final model [35]. However, unlike Stagewise, the LARS algorithm does not do this by taking thousands of tiny steps and recalculating the correlation after each step. LARS pre-computes the size of step it can take in the direction of the predictor d_j prior to another predictor d_i becoming as correlated to the residual. Then, like Stagewise, but unlike Forward Selection, it moves to this point and instead of continuing along d_j, it proceeds along an equiangular direction

Algorithm 2.1 Least Angle Regression [35].

Input: $x \in \Re^m$ (standardized to have mean zero), $D \in \Re^{m \times k}$ (standardized to have means zero and unit ℓ_2 norm on each column)
Output: $\alpha \in \Re^k$ (coefficient vector)

function LARS(x, D)
  $\lambda \leftarrow$ (Regularization parameter value)
  $\alpha \leftarrow$ (Zero vector of k), $\mu \leftarrow$ (Zero vector of m)
  $\mathcal{A}$ (active set), $\mathcal{A}^c$ (inactive set)
  $D_{\mathcal{A}}$ (is a matrix with active set elements from D)
  $signOk \leftarrow true$, $nVars \leftarrow 0$
  while $nVars < min(k, m)$ **do**
    $\hat{c} \leftarrow D^T(x - \mu)$ ▷ vector of current correlation for each element in D
    $\{\hat{C}, \hat{j}\} \leftarrow \text{argmax}_{j \in \mathcal{A}^c}\{abs(\hat{c})\}$ ▷ most correlated vector of the inactive set
    if $signOk$ **then**
      Update $\mathcal{A}$, $\mathcal{A}^c$ so that $\mathcal{A} = \mathcal{A} \cup \{\hat{j}\}$
      $nVars+ = 1$
    end if
    $S_{\mathcal{A}} \leftarrow sign(\hat{c}(\mathcal{A}))$ ▷ sign vector of correlation for active set elements
    $\delta \leftarrow \frac{1}{\sqrt{S_{\mathcal{A}}^T (D_{\mathcal{A}}^T D_{\mathcal{A}})^{-1} S_{\mathcal{A}}}}$
    $w_{\mathcal{A}} \leftarrow \delta (D_{\mathcal{A}}^T D_{\mathcal{A}})^{-1} S_{\mathcal{A}}$ ▷ $w_{\mathcal{A}}$ is a k—vector that is 0 outside of the active set
    $u_{\mathcal{A}} \leftarrow D_{\mathcal{A}} w_{\mathcal{A}}$ ▷ vector equiangular to elements of $\mathcal{A}$
    $a \leftarrow D^T u_{\mathcal{A}}$ ▷ vector of angles between d_j and $u_{\mathcal{A}}$
    if $nVars == min(k, m)$ **then**
      $\gamma \leftarrow \frac{\hat{C}}{\delta}$ ▷ step toward most correlated element
    else
      $\gamma \leftarrow \text{argmin}^+_{j \in \mathcal{A}^C} \left\{ \frac{\hat{C} - \hat{c}_j}{\delta - a_j}, \frac{\hat{C} + \hat{c}_j}{\delta + a_j} \right\}$ ▷ find smallest positive step
    end if
    $\alpha_{tmp} \leftarrow \alpha_{\mathcal{A}} + \gamma w_{\mathcal{A}}$ ▷ updates coefficients corresponding to the elements of $\mathcal{A}$
    $signOk \leftarrow true$
    $\{\hat{\gamma}, j\} \leftarrow \text{argmin}_{j \in \mathcal{A}} \left(-\alpha_{\mathcal{A}}^T ./ w_{\mathcal{A}}\right)$ ▷ min. of element-wise division on $\mathcal{A}$
    if $\hat{\gamma} < \gamma$ **then** ▷ step larger than allowed, need to update
      $\gamma \leftarrow \hat{\gamma}$ ▷ set new step size
      $\alpha_{tmp} \leftarrow \alpha_{\mathcal{A}} + \gamma w_{\mathcal{A}}$ ▷ update α_{tmp} for new step size
      $\alpha_{tmp} \leftarrow \alpha_{tmp} - \{j\}$ ▷ remove offending coefficient
      $nVars- = 1$ and $signOk \leftarrow false$
    end if
    $\mu \leftarrow \mu + \gamma u_{\mathcal{A}}$ and $\alpha \leftarrow \alpha_{tmp}$
    if $((mode == \ell_1) \& (\ell_1(\alpha) > \lambda)) \| ((mode == \ell_2) \& (\ell_2(x - \mu) < \lambda))$ **then**
      break
    end if
  end while
return α (coefficient vector)
end function

between d_j and d_i until the next predictor is as correlated as the first two. This process is repeated until the desired k-parameter linear model is formed [35].

Many of the details of the LARS process presented in Algorithm 2.1 have been omitted here for brevity, such as how the LARS process is modified to produce the Lasso by requiring the sign of the coefficients in the coefficient vector α to stay the same in a given step. Additional details on how to calculate the length of a step along a direction before the next predictor becomes equally correlated with the residual, or how the equiangular direction between two predictors d_j and d_i is calculated, have also been omitted here. We direct the interested reader to [35] for more information on these subjects.

Before moving on to the discussion of dictionary learning, it should be noted that there are two different stopping criteria in Algorithm 2.1, which are used in different circumstances. When $mode == \ell_1$, the stopping criteria is based upon the ℓ_1 norm of the coefficient vector; once the norm reaches a user defined threshold λ, or all of the coefficients are used, the algorithm terminates. This is used once the dictionary is learned and we wish to sparsely represent the input signal x as the linear combinations of a few dictionary elements. The second stopping criteria is when $mode == \ell_2$. In this case, the algorithm terminates when the ℓ_2 norm of the residual falls below the λ threshold, or all of the coefficients are in use. In other words, termination occurs when the ℓ_2 error of the regression model has fallen below the user defined λ value. This is used while learning the dictionary as we wish to have dictionaries that reproduce the input signal x with the least possible error.

2.2.3 *Dictionary Learning*

The learning of a basis dictionary $D \in \Re^{m \times k}$, where k is the chosen number of basis vectors to learn, is the next step in the process of constructing sparse coded vectors for a set of input signals. The extraction process previously described in Section 2.2.1 creates the set of input column vectors of the signal matrix, X, which are used as input

Algorithm 2.2 Online Dictionary Learning [31].

Input: $p(x)$ (An algorithm to draw independent identically distributed (i.i.d.) samples of vectors $x \in \mathfrak{R}^m$ from $\mathcal{X}$)
Output: $D \in \mathfrak{R}^{m \times k}$ (learned dictionary)

function DICTLEARN($p(x)$)
  $\lambda \leftarrow$ (Regularization parameter value)
  $T \leftarrow$ (max number of iterations)
  $D_0 \in \mathfrak{R}^{m \times k} \leftarrow$ (initialize by drawing from $p(x)$)
  $A_0 \in \mathfrak{R}^{k \times k} \leftarrow 0$
  $B_0 \in \mathfrak{R}^{m \times k} \leftarrow 0$
  for $t = 1$ to T **do**
    $x_t \leftarrow$ i.i.d sample from $p(x)$.

$$\alpha_t \triangleq \underset{\alpha \in \mathfrak{R}^k}{\operatorname{argmin}} \frac{1}{2}\|x_t - D_t\alpha\|_2^2 + \lambda\|\alpha\|_1$$

▷ Coding: using Least Angle Regression [35]

    $A_t \leftarrow A_{t-1} + \alpha_t \alpha_t^T$
    $B_t \leftarrow B_{t-1} + x_t \alpha_t^T$
    $D_t \leftarrow \text{DICTUPDATE}(D_{t-1}, A_t, B_t)$ ▷ Algo. 2.3
    ▷ **So That:**

$$\begin{aligned} \alpha_t &\triangleq \underset{D \in C}{\operatorname{argmin}} \frac{1}{t} \sum_{i=1}^{t} \left(\frac{1}{2}\|x_i - D_t\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1 \right), \\ &= \underset{D \in C}{\operatorname{argmin}} \frac{1}{t} (\text{Tr}(D^T D A_t) - \text{Tr}(D^T B_t)) \end{aligned} \tag{2.5}$$

  end for
return D_T (learned dictionary)
end function

to the dictionary learning process. Though the number of samples in $\mathcal{X}$ change with the size of the area of interest and the step size when sliding the extraction window, the dictionary learning process is conducted in the same way. This is to say that the size of the dictionary is not dependent upon the sample size in $\mathcal{X}$, though more samples tend to produce better dictionaries capable of spanning the entire input set. Note that although the dictionary size does not change with the number of samples, the dictionary size will change with the length of the column vectors $x \in \mathcal{X}$, which are dependent upon the size

of the sliding window of size p by p and the number of parameters being concatenated. As the m in signal vector $x \in \mathfrak{R}^m$ increases, so does the m in dictionary $D \in \mathfrak{R}^{m \times k}$, since we will be using the linear combination of the elements of D to reconstruct the signal vectors later.

For this work, we have chosen to utilize the dictionary learning algorithm of [31], which is shown in Algorithm 2.2. In the classical problem of dictionary learning for sparse representation of a finite set of signals $\mathcal{X} = [x_1, \ldots, x_n]$ where $x \in \mathfrak{R}^m$, the underlying objective is to optimize the empirical cost function:

$$f_n(D) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, D) \tag{2.6}$$

where D in $\mathfrak{R}^{m \times k}$ is the dictionary. Each column of D represents a basis vector, where ℓ is some loss function that decreases to some arbitrarily small value when the dictionary D represents the signal x in a sparse fashion, and with little error [31].

In this problem, the number of samples n is usually large, whereas the signal dimension m is relatively small. For example, $m = 64$ for an image patch sized 8 by 8 cells, and $n \geqslant 10,000$. It is also generally assumed that we have $k \ll n$ (e.g., $k = 200$ for $n = 100,000$), and each signal only uses a few elements of D in its representation. Note that in this setting, overcomplete dictionaries with $k > n$ are also allowed.

The loss function $\ell(x, D)$ is usually defined as the optimal value of the ℓ_1 sparse coding problem:

$$\ell(x, D) \triangleq \underset{\alpha \in \mathfrak{R}^k}{\operatorname{argmin}} \frac{1}{2} \|x - D\alpha\|_2^2 + \lambda \|\alpha\|_1 \tag{2.7}$$

where λ is a regularization parameter. This problem is also known as basis pursuit or the least absolute shrinkage and selection operator (Lasso). It is well known that ℓ_1 regularization yields a sparse solution for α, but there is not a direct analytical link between the value of λ and the corresponding effective sparsity $\|\alpha\|_0$. In order to prevent

D from having arbitrarily large values (which, in turn, leads to arbitrary small values of α), the columns of D $(d_1, \ldots, d_k)$ are constrained to have an ℓ_2-norm less than or equal to one [31]. The convex set of matrices that satisfy this constraint are labeled C and are defined as:

$$C \triangleq \{D \in \Re^{m \times k} \text{ s.t. } \forall\, j = 1, \ldots, k,\ d_j^T d_j \leqslant 1\} \tag{2.8}$$

In Algorithm 2.2, this is the set of matrices that dictionary D is required to be an element of after each update (call to Algorithm 2.3). Note that this constraint also satisfies a constraint in Algorithm 2.1, requiring that the columns of predictor variables (columns of D) have a unit ℓ_2-norm. Though the problem of minimizing the empirical cost $f_n(D)$ is not convex with respect to D, it can be rewritten as a joint optimization problem with respect to the dictionary D and the coefficients $\mathcal{A} = [\alpha_1, \ldots, \alpha_n]$ in $\Re^{k \times n}$ of the sparse decompositions. This joint optimization problem is not jointly convex, but convex with respect to each of the two variables D and $\mathcal{A}$ when the other one is fixed:

$$\min_{D \in C,\ \mathcal{A} \in \Re^{k \times n}} \sum_{i=1}^{n} \left(\frac{1}{2}\|x - D\mathcal{A}\|_2^2 + \lambda \|\alpha_i\|_1\right) \tag{2.9}$$

The optimization method that we use was developed in [31] and is in a class of online algorithms based on stochastic approximations, processing one sample at a time making a sequence of updates to D

$$D_t = \prod_C [D_{t-1} - \delta_t \bigtriangledown_D \ell(x_t, D_{t-1})] \tag{2.10}$$

where D_t is the estimate of the optimal dictionary at iteration t, δ_t is the gradient step, $\prod_C$ is the orthogonal projector onto C, and the vectors x_t are independent identically distributed (i.i.d) samples of the (unknown) distribution $p(x)$ of the data.

Algorithm 2.3 Dictionary Update [31].

Input: $D = [d_1, \ldots, d_k] \in \Re^{m \times k}$ (input dictionary)
Input: $A = [a_1, \ldots, a_k]\Re^{k \times k}$
Input: $B = [b_1, \ldots, b_k]\Re^{m \times k}$
Output: $D \in \Re^{m \times k}$ (updated dictionary)

function DictUpdate(D, A, B)
  $k \leftarrow$ (Max iterations for update)
  repeat
    for $j = 1$ to k **do** ▷ Update the j^{th} column to optimize for Equation 2.5 in Algo. 2.2
      $d_j \leftarrow$ (j^{th} column of D)

$$u_j \leftarrow \frac{1}{A[j,j]}(b_j - Da_j) + d_j$$
$$d_j \leftarrow \frac{1}{\max(\|u_j\|_2, 1)} u_j \tag{2.11}$$

    end for
  until convergence ▷ until $\ell_2(D)$ stops changing by more than $1e^{-15}$
return D (updated dictionary)
end function

This algorithm operates on the premise that one is not interested in the minimization of the empirical cost function $f_n(D)$ with a high degree of precision. Instead, it is assumed that the minimization of the expected cost of the function is more desirable, which is taken relative to the (unknown) probability distribution $p(x)$. The expectation is that one should not spend a lot of effort on the accurate minimization of the empirical cost, since it is but an approximation of the expected. As such, "inaccurate" solutions may provide similar, if not better, cost than one that attempts to gain a higher degree of accuracy [31].

The dictionary learning process assumes that the training set is composed of i.i.d. samples of a distribution $p(x)$. However, as was mentioned by [31], it is often difficult to obtain such i.i.d. samples, so the input dataset is obtained as a cycle on a random permutation of the sample matrix $\mathcal{X}$. So, each of the T iteration of the algorithm starts by drawing an element x_t from the random permutation of sample matrix $\mathcal{X}$. Then, x_t is

used to compute α_t of x_t over the dictionary D_{t-1}, which was obtained from the previous iteration. The dictionary D_{t-1} is updated to be the new dictionary D_t in Algorithm 2.3 by minimizing over C, the function

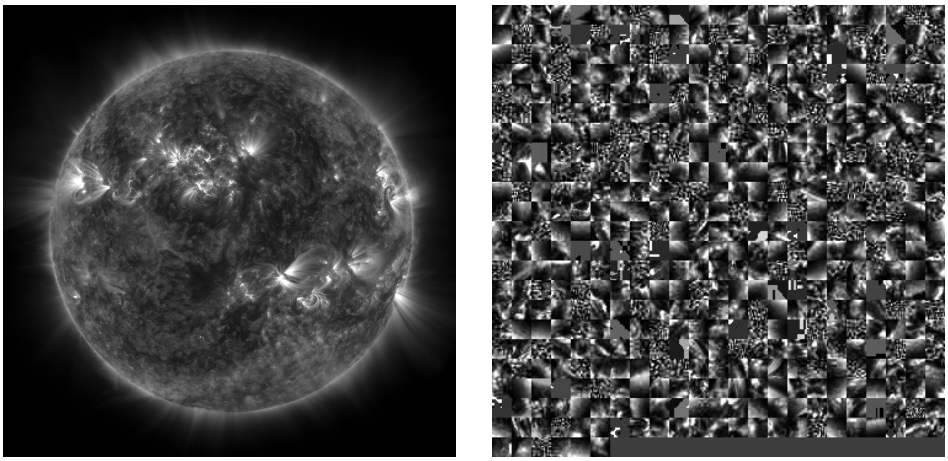
$$\hat{f}_t(D) \triangleq \frac{1}{t}\sum_{i=1}^{t}\left(\frac{1}{2}\|x_i - D_t\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1\right) \tag{2.12}$$

where the vectors α_i for each $i < t$ were computed during the previous steps of the algorithm, and stored as updates to A_{t-1} and B_{t-1} in such a way that previous updates have less influence than current updates.

Since each column of D_{t-1} is updated independently, by solving Equation 2.5 with respect to the j^{th} column d_j, and keeping the other columns of D_{t-1} fixed under the constraint $d_j^T d_j \leqslant 1$, the update is equivalent to an orthogonal projection of the vector u_j defined in Equation 2.11, onto the constraint set [31]. This guarantees the update dictionary D_t is a member of the set C.

The exact number of basis vectors (k) that we have chosen to use varies by application and experiment, and will be discussed more in the appropriate sections dealing with each application area. However, when using multiple image parameters from multiple wavebands of images, we generally choose $k < m$ to create a smaller dictionary dimension than our input dimension, effectively performing dimensionality reduction of the extracted, overlapping, p by p cell windows.

In Figure 2.4(b), we present an example of a learned dictionary from image patches extracted directly from the pixel values of a 171Å waveband image shown in Figure 2.4(a). This dictionary is used to then recreate the original image from a set of sparse coding vectors in Figure 2.4(d). Since the mean of the signal vector must be subtracted from the input to the LARS algorithm, we also show the values that are added back to the reconstructed patch in Figure 2.4(c). Though these figures were originally constructed for functional verification, they also show how the dictionary learning process produces



(a) Original 171Å Image

(b) Learned Dictionary

(c) Patch Pixel Means

(d) Recreated 171Å Image

Figure 2.4: An image from AIA 171Å(2.4(a)), with an example of the dictionary learned from it (2.4(b)), and the reconstruction of that image (2.4(d)) from a set of sparse vectors and the mean values subtracted from the input signals (2.4(c)).

a set of higher level concepts (line segments, light and dark areas, etc.) that are used to describe a location on an image. It is by using these higher level concepts that we will discern how similar regions of images and whole images are. We will also show how they can do this better than simply using low level comparisons such as pixel level information.

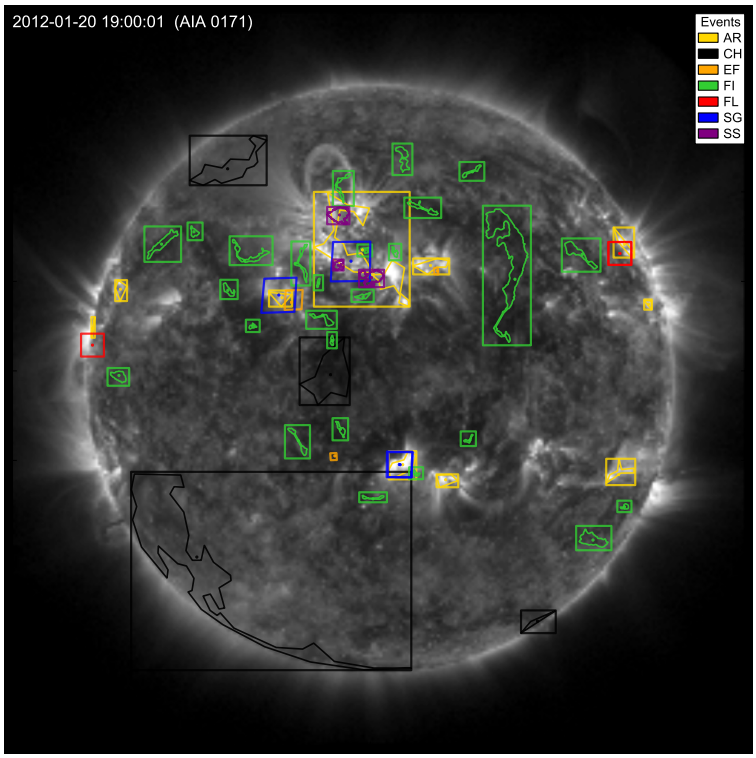


Figure 2.5: An image from the SDO mission, with labeled solar phenomena coming from HEK overlaid. Labels include (AR) Active Region, (CH) Coronal Hole, (EF) Emerging Flux, (FI) Filament, (FL) Flare, (SG) Sigmoid, and (SS) Sunspot.

2.3 Tracking Solar Events

The task of tracking solar events was the first application area that we applied the dictionary learning and sparse vector representation to. In this task, we needed to link together the detections of various solar activity that came from work that was done prior to the launch of the SDO mission. In this prior work, a multi-institutional team called the Feature Finding Team (FFT) created a number of software modules to process data coming from the SDO. These modules create reports of solar phenomena (events) that are of great interest to the solar physics research community. As such, the metadata they create is directly reported to the Heliophysics Event Knowledgebase (HEK) [36], and is available to the general public through the online iSolSearch graphical interface[4]. An example of both the image data produced by SDO's Atmospheric Imaging Assem-

4 https://www.lmsal.com/isolsearch

bly (AIA), as well as the metadata reported to HEK can be seen in Figure 2.5. Some of the reported events, such as Active Regions (AR), can be seen in the wavelength being displayed, while others, such as Sunspots (SS), are only visible in other wavelengths.

The vector data from the HEK, which we utilize for tracking, contains at minimum, the following four fields for each detection report:

- *Identifier:* a unique identifier
- *Time Stamp:* the starting time
- *Center Point:* the center of the detection
- *Minimum Bounding Rectangle:* a set of four points that produces the minimum-sized rectangle that encompasses the detected object.

Some event types also contain an additional field if the detection module produces the information:

- *Boundary Polygon:* a set of points that constitutes the polygon representation of the boundary of the detected object. Some event types do not report a polygon representation, leaving this field null for those types of events.

If the Boundary Polygon field is not present, we copy the Minimum Bounding Rectangle filed as a polygon so as to make processing simpler in our tracking algorithms. Finally, in order to identify which detection belongs to which track, we also added the following field:

- *Next Detection Identifier:* the unique identifier of the next detection of the same type of solar event that is part of the same trajectory, if such a detection report exists. This field is null if there is no next event detection in the trajectory.

These vector objects are in Helioprojective Cartesian (HPC) coordinate system, which is converted to a pixel-based coordinate system in our algorithm so that the detections' coordinates corresponded with the proper location on the raster data.

In the initial work that we produced for tracking solar events in [17] and [18], a tracking algorithm was developed in order to produce various tracked solar events from input detections reported to the HEK. The primary purpose of the output from this tracking algorithm was to facilitate the spatiotemporal analysis of the data reported to the HEK by the FFT modules. Spatiotemporal analyses, such as spatiotemporal co-occurrence pattern mining [37], require not just a spatial representation of an object at a given time step, but also the information about its evolution over time. Spatiotemporal co-occurrence pattern mining utilizes this information to find evolving events that frequently occur in proximity to one another. Similarly, spatiotemporal event sequence pattern mining [38] utilizes the same input to find frequently occurring patterns of events leading to other events.

One of the crucial components of such tracking algorithms is an appearance model that is able to discriminate among different targets. However, when using a fixed model based on the distribution of histogram distances between in-group and out-group regions, as was done in [17, 18], global appearance information about each type of target must be obtained before it can be utilized to discriminate between the same object at a later time, and another tracked object in a similar location. This type of global learning can only be accomplished when there is prior knowledge of tracked solar events. This knowledge can be obtained by human labeling of event tracks, from tracked solar events produced by another tracking algorithm, or, as was done in [17, 18], with the use of heuristics to link small sets of detections into shorter tracks.

An improvement upon this fixed model can be achieved through the use of an online sparse coding and dictionary learning model. In the online model, the appearance of the object being tracked is learned by the dictionary learning process already described in Section 2.2.3, and then possible matching candidates are compared to find the most probable matching event report. With such an online model, the need for solar events to be tracked for appearance model learning is eliminated, which makes it simpler to apply

the tracking algorithm of [17, 18] to solar event types without this information. In [16], we developed this online sparse coding dictionary learning model, and will discuss it in detail in Chapter 4, along with additional information about how the appearance model is used in the tracking process.

2.4 Image Retrieval on Solar Images

After the use of dictionary learning and sparse vector representation for tracking solar activity, we pursued other uses for this form of image representation and comparison. Namely, we investigated how to use these methods to create a system of content based image retrieval (CBIR) on the SDO AIA images. In [39], a 64-bin histogram for each of the 10 parameters extracted from the image was utilized in an attempt to produce a CBIR system. The examinations of several dissimilarity metrics in [39] found that the most similar images were almost exclusively found to be from the preceding or succeeding time step(s), and a few possible solutions for these phenomenon were explored. One possible solution was to increase the time between samplings of the dataset. However, as was noted by [39], this leads to some events in the data being completely missed, as their lifespans are shorter than the sampling period. Eventually, the similarity of temporal neighbors problem was left as an open issue to be addressed in future work, and the dimensionality of the descriptors preventing indexing was never addressed.

Later, [29] revisited investigating a CBIR system, this time for region retrieval using statistical descriptors by calculating a 7-statistic summary as the descriptor of each event region on each image parameter. These statistics were: *minimum*, 1^{st} *quartile*, *median*, 3^{rd} *quartile*, *maximum*, *average*, and *standard deviation*. The descriptor statistics were ranked using an F-Statistic ranking method similar to what we will describe in Chapter 3, and the top-N were chosen to represent the labeled regions. This provided a smaller, indexable descriptor for the labeled regions; these results will be the benchmark to which we compare our own results later in Chapter 6.

During our work on tracking solar events, we implemented a method based on sparse coding in order to differentiate between different classes of the same type of solar event in SDO AIA images in [16]. In that work, it was shown that the use of sparse coding was more selective than histograms of image parameters in the task of differentiating between solar events of similar structure. This ability to differentiate between regions of the same type of solar event in these images was the motivation for our preliminary investigation into using a sparse coding method to describe whole solar SDO AIA images for similarity search in [40]. Our methods of using sparse coding for image retrieval focus on addressing the deficiencies of previous methods for comparing whole images and regions of images as described above, such as the inability to index or only finding temporal neighbors when querying for similar images. More details of both the whole image and region comparison methods for SDO AIA image CBIR will be provided in Chapter 5.

3 FEATURE SELECTION

Before presenting the first use case of tracking solar events using the previously outlined sparse coding methods, it will be useful to give a thorough account of how a subset of the available image parameters is chosen for our applications where we are not utilizing the entire feature space. In the tracking use case, the goal is to identify the most characterizing features of the observed solar event data from the image parameter data, i.e., feature selection. It is well known that feature selection is a critical step used to select a subset of the input for use in classification tasks, and there are two main advantages to utilizing feature selection, which are: (1) the reduction of the number of dimensions in the problem and therefore a reduction of the computational cost of classification; (2) reducing noise in the input data and subsequently increasing classification accuracy.

The formal definition of feature selection can be described as follows: Given the input data D as a set of N samples and M features $X = \{x_i, i = 1, \ldots, M\}$, and the target classification variable c, the problem of feature selection is that of finding from the $M-$ $dimensional$ observation space, $\Re^M$, a subspace of m features, $\Re^m$, which "optimally" characterize c, where $m \ll M$ [41]. Less formally, our goal for solar event tracking is to find a subset of image parameters from the set of 90 image parameters (9 wavebands of images at each timestep $\times$ 10 parameters calculated for each image) that best highlights the similarity of two consecutive reports of the same solar event and differentiate them from reports of similar solar activity in the same period.

In general, there are two approaches to standalone feature selection: filters and wrappers [42]. In the filter type of feature selection, features are selected based upon some intrinsic characteristic of the feature that determines its power to discriminate between the target classes. We have chosen the F-Statistic as the proxy measure for discrimination

power of our features. We base this choice on the fact that it is fast to compute and was found by [43] to still capture the usefulness of the features being evaluated. The filter type of feature selection methods have an advantage of being relatively easy to compute and the characteristics are uncorrelated to the learning methods that will be applied to the data, which leads to better generalization. However, filter methods can sometimes tend to select redundant features because they do not consider the relationships between features. In the wrapper type of feature selection, the selection of features is "wrapped" around some learning method, and the usefulness of features is directly evaluated by the estimated accuracy of the learning method. This method, however, potentially entails significant computational cost in the search for an optimal set of features.

In the following sections of this chapter, we present three different feature selection methods that we evaluated for use in choosing a subset of the available image parameters from our dataset. The work was presented in [44], and is used to justify our selection method for tracking and image retrieval tasks. The first selection method, discussed in Section 3.1, is a simple filter type that selects the Top-K features based upon some ranking criteria. The second method, described in Section 3.2, is a hybrid filter and wrapper method that first utilizes the ranking of features and then utilizes a learning method to evaluate the addition of each ranked result. Finally, the third method, presented in Section 3.3, is a more complex filter type that again utilizes the same ranking of features that the first two do, but then also uses a measure of redundancy of the features to exclude some of the ranked results. We then describe how we utilize these methods on our solar events and provide experimental evaluations of each method in Section 3.4.

3.1 Top-K Rank

In this filter method of feature selection, we select the K top-ranked features, where the ranking method looks to order the features by their relevance to the target class c. Here we have chosen the F-Statistic between the features and the classification variable as the

scoring method used for the ordering of features. In this, the F-Statistic of feature x_i in P classes denoted by c is as follows:

$$F(x_i, c) = \frac{\sum_{j}^{P} n_j(\bar{x}_{ij} - \bar{x}_i)^2/(P-1)}{\sigma^2} \tag{3.1}$$

where $\bar{x}_i$ is the mean value of feature x_i across all classes, $\bar{x}_{ij}$ is the mean value of x_i within the j^{th} class, n_j is number of samples of x_i within in the j^{th} class, P is the number of classes, and

$$\sigma = [\sum_{j}^{P}(n_j - 1)\sigma_j^2]/(n - P) \tag{3.2}$$

is the pooled variance across all classes (where σ_j is the variance of x_i within the j^{th} class) [42]. Using the value returned by the F-Statistic as the relevance of the tested feature, we order the features by descending value and choose the Top-K.

3.2 Top-K Forward Selection

This method combines both the filter and wrapper methods of feature selection. The steps of this method are as follows:

1. Rank features by F-Statistic as was done in Top-K method.
2. Evaluate classification accuracy of the top ranked feature.
3. Add next highest ranked feature to the subset and evaluate accuracy.
4. If accuracy increases, keep the feature in the subset; otherwise remove it.
5. If we reach the number of features we want, then stop, otherwise repeat starting from step 3.

This selective addition method evaluates the classification accuracy of the feature subset by using a Naïve Bayes' classifier as a simple learning method wrapper.

3.3 Top-K Redundancy-Limiting Forward Selection

This method is similar to the previous method in that feature subset $\bar{X}_i$ is obtained by adding the next highest ranked feature not already attempted, from the ranked list of all features, to the feature subset $\bar{X}_{i-1}$; the resultant feature set $\bar{X}_i$ is evaluated using some selection criteria. However, instead of utilizing a learning method and evaluating if the addition increases classification accuracy as the selection criteria, this method utilizes a redundancy limiting condition. The redundancy value is calculated using the Pearson's correlation coefficient $cor(x_i, x_j)$ as follows:

$$R_i = \frac{1}{(|\bar{X}_i| * (|\bar{X}_i| + 1))/2} \sum_{i,j} |cor(x_i, x_j)| \tag{3.3}$$

Where $|cor(i, j)|$ is the absolute value of the Pearson's correlation coefficient between the i^{th} and j^{th} feature. The feature added to subset $\bar{X}_i$, at each step, is kept if it does not increase the redundancy (R_i) of the feature subset by some threshold amount, say 5% over the redundancy R_{i-1} of the previous set $\bar{X}_{i-1}$.

3.4 Feature Extraction and Feature Ranking

The definitions for the three feature selection methods above utilize a feature value x_i by making the assumption that this feature is a variable of constant length one. However, in our implementation, each instance of an event report contains several cells of an image parameter inside its boundary, which is not a consistent size from one detection to the next. Thus, to compare two event reports and get a single feature value x_i, we begin by extracting a 20-bin histogram of the cells that fall within the bounding box of an event

report. Then, this process is repeated for a second report for subsequent comparison with the first report. Once the two histograms are created, we then compare them using a common histogram comparison method and use the resultant value of the comparisons as our feature value x_i. The histogram comparison method that we use for this is the *histogram intersection*, where intersection is defined as the sum of the minimum bin count between the two histograms being compared or:

$$d(H_1, H_2) = \sum_{I}^{N} min(H_1(I), H_2(I)) \tag{3.4}$$

Where $H_k(I)$ is the value of the histogram bin at index I of N bins in histogram k. This was chosen as an approximation of the amount of mutual information between the two histograms and we found it to best generalize for ranking through testing four different histogram comparison methods (Correlation, Chi-Squre, Intersection, and Bhattacharyya distance). Note that this histogram construction and comparison is done on a per image parameter and waveband basis, meaning that we construct a new variable x for each of the 90 parameter/waveband combinations.

To produce a feature value for one class, the feature ranking process requires two event detections. A histogram of parameter values is created for each detection, and then feature value x_i is produced by use of the histogram intersection. In order to produce feature value x_i for the P^{th} class, this process is repeated by keeping the first detection and replacing the second with one from the P^{th} class and calculating the histogram intersection of the new pair. This replacement is done until there is a value of x_i for each of the P classes, then a new first detection is chosen so that the feature value x_{i+1} can be calculated. This continues until the desired number of samples are produced.

The way that detections are grouped into classes is different depending upon the application, and the specifics unique to those applications are discussed in their respective chapters. For the evaluation of various selection methods presented in this chapter, we

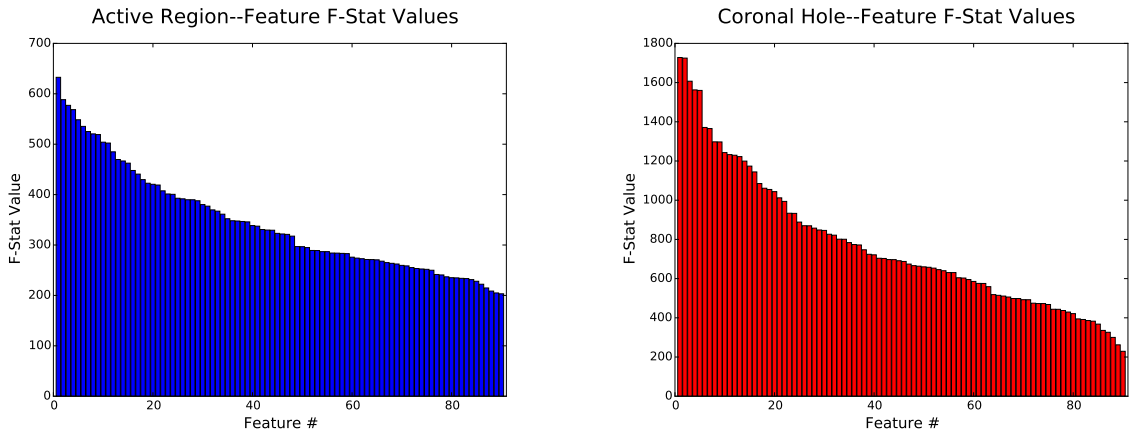
utilize already tracked solar events of type Active Region (AR) and of type Coronal Hole (CH). We have chosen to use the events reported for the two-year period of 1 January 2012 through 31 December 2013, and have split the tracks of events into the respective months in which they occur. This creates 24 independent datasets for us to use.

The detection pairing described above begins by selecting only tracks from one month that have more than one detection in their trajectory. Then, for each pair of detections in the temporal sequence formed by these tracked events, a feature value x_i is produced using the detections within a single track as a comparison point belonging to class 1 (same object). With the first detection chosen, the feature value x_i for class 2 (different object) is produced by comparing it to another detection from the dataset that is known to be from another tracked event trajectory. This process results in a balanced 50/50 split between the two classes. It is these sets that are used in the three feature ranking processes described previously.

To ensure that our evaluation of feature ranking was not biased by the month that we selected as the ranking data, we repeated the ranking of the features using several different months. In repeating the ranking, we saw that the top 10 ranked features stayed within the top 20 ranking and generally moved by only a few positions. We also saw that the top-ranked feature for each event type generally stayed in the top 3 positions across all the tested months.

3.4.1 Experimental Framework

In order to test the ranking methods we wish to evaluate, we use how well we can classify between the two classes of same object and different object, as described in the previous section, when using the top-ranked features of each method. The classification performance is evaluated using the "Leave-One-Out Cross Validation" (LOOCV) method. In order to perform the cross validation, we took the 24 months of data that we had available and set one month aside for testing, while training on the remaining 23. In the



(a) (b)

Figure 3.1: 3.1(a) Active Regions, 3.1(b) Coronal Holes: F-Statistic values of all 90 features ordered by their F-Statistic value.

training process, we used a similar feature extraction method of comparing histograms from the same trajectory and different trajectories as was used in the ranking process. The major difference being that we use 23 months of training data to randomly select the event report from the "different object" class in our evaluations, as opposed to the use of only one month in ranking. In the task of classification, we consider an object to be classified properly if the classifier gives the detection from the same tracked object a higher probability of being in the class it belongs to than it does for a second detection from a different tracked object. This choice is based on the fact that, in the tracking algorithm of [18], the path a tracked object takes is affected more by this difference than by whether the object from the same path is more strongly associated with the incorrect class than it is with the correct one.

In our experimental evaluation, we utilized a different histogram distance measure than was used for ranking. We chose to use the Bhattacharyya coefficient instead of the histogram intersection to show that the wavelength and parameter pair features obtained from the selection methods are effective features regardless of feature comparison method. The Bhattacharyya coefficient is an approximate measure of the amount of overlap between two statistical samples which approximates the chi-square measure statistic

for small distances. However, the Bhattacharyya coefficient does not have the drawback of infinite values arising when comparing empty histograms as can happen with the chi-squared statistic [45].

We compare the Top-K ranking selection method against the wrapper method of Top-K Forward Selection and the redundancy limiting selection method of Top-K Redundancy Limiting Forward Selection. The selected features are fed to a Naïve Bayes' classifier and tested against the testing month.

3.4.2 *Results*

We begin our discussion of results with Figure 3.1. In it, we can see how the F-Statistic values for each of the features compares to the rest of the features in our dataset. As can be seen from the charts, the top 10 to 20 features indicate a significant separation of feature values between the two classes, while the latter features are not as cleanly separated. This gives us a reason to suspect that we shall see our best improvements in classification results with the inclusion of the first 10 to 20 features in our models, and modest or even declining accuracy as we include more.

Next, in Table 3.1, we include the top 10 ranked features for each event type used in this evaluation. In this table, it can be seen that the features of most importance, as determined by F-Statistic ranking, are from wavelengths that are not used in the detection of these solar phenomena. For instance, the wavelength that is used in the detection of these phenomena [46] (193Å) does not show up in the ranking until the 10^{th} position in the list for Active Regions, and does not show up at all in the top-10 ranked features for Coronal Holes. This may indicate some heretofore unrecognized information about these event types contained in wavelengths other than those used for detection of these particular event types.

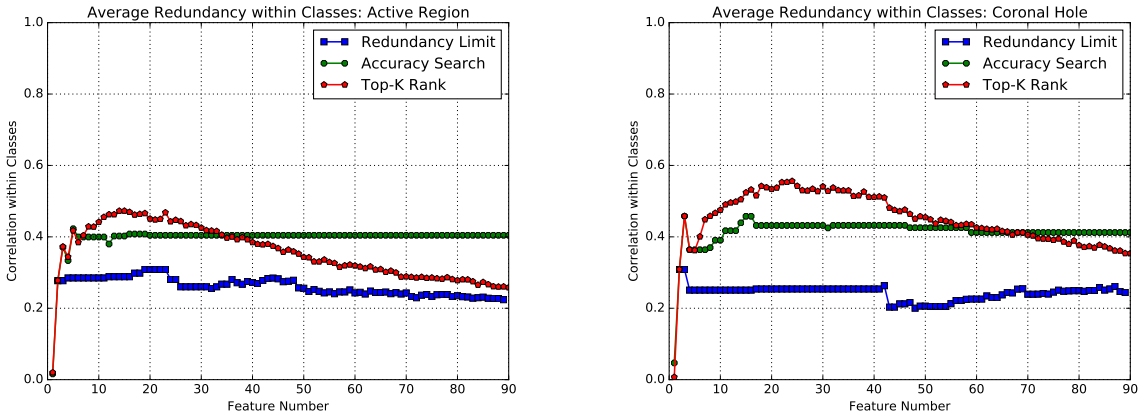
In Figures 3.2, 3.3, and 3.4 we display the progression of our three feature selection methods when we do not limit the number of features they return. The feature numbers

Table 3.1: Top 10 ranked wavelength/parameter pairs for Active Regions and Coronal Holes using F-Statistic values.

Parameter #	Active Regions		Coronal Holes	
	Wave	Param	Wave	Param
1	335Å	Mean	1600Å	Mean
2	1700Å	Mean	1700Å	Mean
3	1600Å	Mean	1700Å	Entropy
4	1700Å	Entropy	1600Å	Standard Deviation
5	211Å	Mean	1600Å	Entropy
6	304Å	Mean	304Å	Mean
7	94Å	Mean	1700Å	Skewness
8	335Å	Entropy	1700Å	Standard Deviation
9	171Å	Mean	1600Å	Skewness
10	193Å	Mean	304Å	Entropy

range from 1 to 90 and are ordered in their ranked order of descending F-Statistic value, though not all features are selected in all methods. Whenever a feature is not selected by a method the score from the previously selected feature is copied to the location in the chart representing the non-selected feature.

We begin with Figure 3.2, which displays the sum of the average redundancy in each class as calculated by Equation 3.3. We display the results for two event types, being Active Regions in Subfigure 3.2(a) and Coronal Holes in Subfigure 3.2(b). As can be seen from the charts, the Top-K selection method ramps up to a maximum around 20 features, which indicates there is a bit of redundancy in the first 20 or so features. However, after the first 20 or so features, the redundancy average begins to decline indicating that the latter features are somewhat less redundant than the previously added features. It can also be seen that the Accuracy Search method does not add all of the first 20 features since the redundant nature of them does not significantly increase the accuracy, hence limiting the total amount of redundancy in the selected feature set. Similarly, the Redundancy Limit method does as it is intended and limits the redundancy in the selected feature set by only adding features that do not significantly increase the total redundancy among the selected features. This would seem to indicate that using Top-K

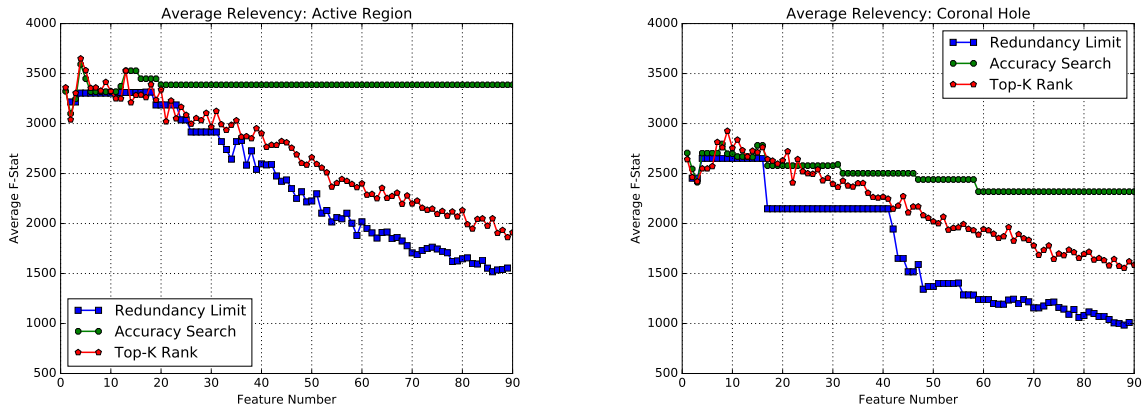


(a) (b)

Figure 3.2: Average Redundancy per feature selected within classes as measured by Equation 3.3. Figure 3.2(a) shows the results for Active Regions, and Figure 3.2(b) shows the results for Coronal Holes.

is too simplistic of a choice for feature selection and that the observed accuracy may be lower than that seen by the other selection methods.

In Figure 3.3, we display the average relevancy of the selected features in the feature set returned by the various selection methods. Again, we provide the results for two different event types, with Active Regions in Subfigure 3.3(a) and Coronal Holes in Subfigure 3.3(b). What is meant by average relevancy in these charts is the average of the F-Statistic value, as calculated by Equation 3.1, for each selected feature. As can be seen from the charts, the Accuracy Search method tends to stop adding features after the first 20 or so, which then keeps the overall relevancy higher than the other two methods. This makes intuitive sense, as it would be sensible to expect that the most relevant features would also be those that produce the best classification results, and simply adding more (but less relevant) features would reduce the accuracy of the classification task. Conversely, in the Redundancy Limit method, the overall relevancy is lower than the other two methods because, as we saw in Figure 3.2, this method does not include all of the most highly ranked features due to the fact that there is a bit of

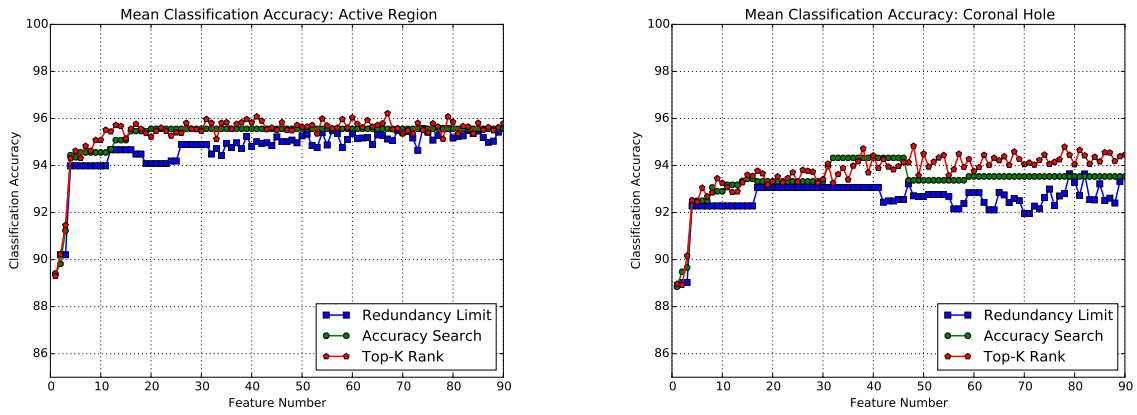


(a) (b)

Figure 3.3: Average Relevancy per feature selected, with relevancy measured as the F-Statistic value of Equation 3.1. Figure 3.3(a) shows the results for Active Regions, and Figure 3.3(b) shows the results for Coronal Holes.

redundancy between these features of higher rank. Note how both the Top-K rank and Accuracy Search seem to track fairly closely up to about 15 features in both event types, which was not the case for the redundancy score. This would seem to indicate that both the Top-K and Accuracy Search methods should see similar accuracy results when using this number of features, even though the former includes more redundant information in its feature set.

Now turning to Figure 3.4, we show the average classification accuracy for all three methods, and again we include both Active Regions in Subfigure 3.4(a) and Coronal Holes in Subfigure 3.4(a). In these figures, we see the classification accuracy rise quickly in the first 10 features and then plateau after about 20 features. This is true for all three selection methods, even though the Redundancy Limit and Accuracy Search methods do not add every feature in the first 20. For instance, the Redundancy Limit method only adds fewer than 10 of the first 40 features of Coronal Holes, as it is trying to keep the added redundancy low. By doing so, it also limits the most relevant features and therefore leads to a lower classification accuracy. As surmised from the relevancy results displayed in Figure 3.3, both Top-K and Accuracy Search perform similarly for the first

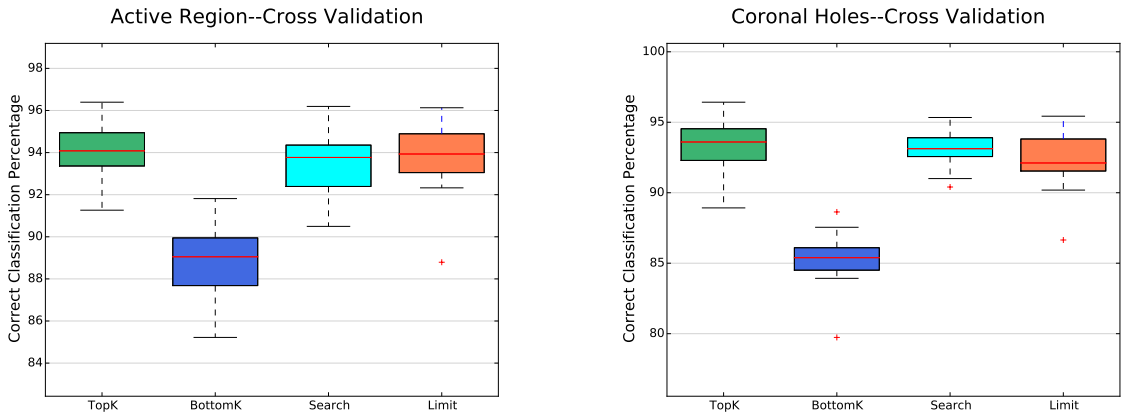


(a) (b)

Figure 3.4: Mean classification accuracy percentage for month following selection month. Figure 3.4(a) shows the results for Active Regions, and Figure 3.4(b) shows the results for Coronal Holes.

15 to 20 added features. However, in the Accuracy Search method on Active Regions, only 10 of the first 20 features are added and then no other features are added from the rest of the dataset as they do not increase the accuracy of classification. This too leads to lower accuracy of classification when compared to the simpler Top-K selection method.

Due to the above observations, we chose to utilize 20 features as a limit for our next set of charts showing the distribution of classification accuracy for the cross validation on our 24 months of data. So, in Figure 3.5, we show the comparison of the cross validation classification accuracy for the three different feature selection methods. Again, we include both the Active Region results in Subfigure 3.5(a) and Coronal Holes in Subfigure 3.5(b). In addition, we have included the cross validation classification accuracy for the bottom 20 ranked features to show how the selected features by each of the methods compared to the lowest ranked features according to our ranking method. As can be seen from the boxplots of Figure 3.5, the Top-K selection method performs as well or better than the other two methods when comparing the classification accuracy by month. In the Active Region plot, the mean of the accuracy distribution for the Top-K method is higher than the other two methods, though the interquartile range is slightly more than



(a) (b)

Figure 3.5: Cross validation accuracy for all 24 months of data. **TopK** uses the top 20 features selected with the Top-K method. **BottomK** uses the bottom 20 features selected with Top-K method just ranked in reverse order. **Search** uses the search method to select up to 20 features. **Limit** uses the limit method to select up to 20 features. Figure 3.5(a) shows the results for Active Regions, and Figure 3.5(b) shows the results for Coronal Holes.

that of the Redundancy Limit method. Similarly for the Coronal Hole plot, the mean is higher on the Top-K method, though the interquartile range is slightly larger than that of the Accuracy Search method.

The results of these experiments have shown us two things. First, a reasonable level of accuracy in classifying two detections as being either the same tracked object or a different tracked object can be achieved with a subset of the dimensionality reducing image parameters. Second, for our dataset of image parameters, and for the task of differentiating between similar regions of solar images, we can use the simpler Top-K feature selection method and arrive at a suitable combination of image parameters from various wavelengths by using a simple method of ranking on the F-Statistic value. The observations that we found when evaluating the ranking position across several months, coupled with the results seen in the classification task, lead us to the conclusion that the simpler method of Top-K selection tends to generalize better than the other selection

methods that we used in this research, and that the top 20 features should be sufficient for tracking, which we will discus in the next chapter.

4 SPATIO-TEMPORAL EVENT TRACKING

As has been mentioned previously, in order to facilitate the important needs of space weather monitoring (which can have vital impacts on space and air travel, power grids, GPS, and communication devices), many software modules developed by the Feature Finding Team (FFT) work continuously on the massive SDO raster data, generating object data with spatiotemporal characteristics. The object data with spatiotemporal characteristics generated by FFT software modules is directly reported to the Heliophysics Event Knowledge (HEK) base [36]. This spatiotemporal data is used to accurately catalog, explore, track, and correlate solar events. To capture correlations between solar events, it is essential to develop tracking techniques that are capable of handling multiple solar event instance hypotheses concurrently.

In our previous work, we began development of a tracking algorithm in [17] and [18] to facilitate the spatiotemporal analysis of the data reported to the HEK by the FFT modules. Spatiotemporal analyses, such as spatiotemporal co-occurrence pattern mining, require not just a spatial representation of an object at a given time step, but also the information about its evolution over time. It utilizes this information to find evolving events that frequently occur in proximity to one another. Similarly, spatiotemporal event sequence

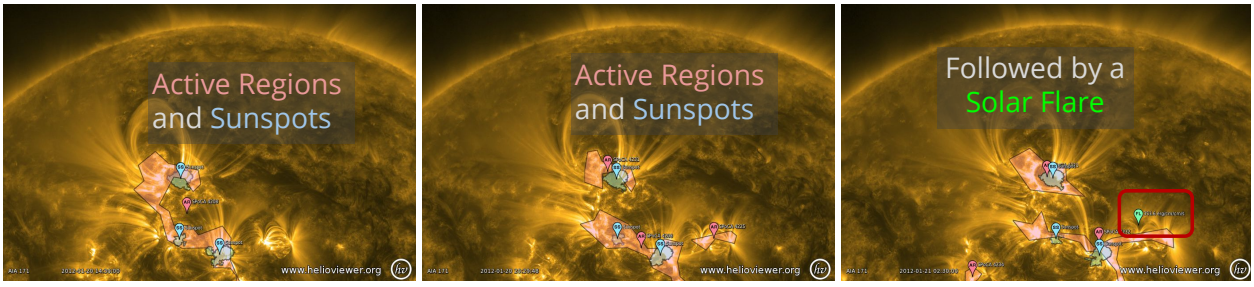


Figure 4.1: An instance of active region co-occurring with a sunspot followed by a solar flare, which occurring between 2012-01-20T14:30:00 and 2012-01-21T02:30:00 [47].

pattern mining utilizes the same input to find frequently occurring patterns of events leading to other events [48].

An example of an application area for spatiotemporal event sequence or co-occurrence pattern mining is in space weather prediction. In Figure 4.1, active regions are shown co-occurring with sunspots, with one of the co-occurrences being followed by a solar flare. Identifying such spatiotemporal event sequences may lead to improved understanding of relationships between differing solar event types, and better modeling and forecasting of important events such as coronal mass ejections and solar flares [38]. These and other spatiotemporal mining algorithms that require spatiotemporal trajectories are discussed in [49], as well as their possible applications to solar data mining.

Though some of the FFT modules produce tracking information, such as the SPoCA module [46], the majority do not. Those that do are part of the detection module and are not meant to be used as a separate general purpose tracker for other event types. Therefore, our goal was to produce a general purpose tracking suite for use across multiple solar event types. With such a general purpose tracking suite, we can produce the spatiotemporal trajectories used in spatiotemporal frequent pattern mining on several event types. This spatiotemporal trajectory production could also include new event types as detection reports become available from new detection modules.

In the previous solar event tracking work of [17] and [18] the visual comparison of event detections relied on learned global distributions of histogram similarity measures, which requires previously tracked data in order to learn the expected distributions of these values. This requirement can be met with some event types that have previously tracked ground truths available, such as active regions and coronal holes. However, there are several other event types where this information is not available, making the use of these learned global distributions of histogram similarity untenable as a method for visual comparison of these event types.

To overcome these limitations, thereby allowing us to perform tracking on event types that do not have previously tracked ground truths, a new appearance model that can be learned online, with no previous knowledge of how the solar events evolve visually over time, was developed in [16]. This online model was shown to perform as well as the previously used offline learned model at the task of differentiating between detections in the same trajectory and detections belonging to different trajectories. Therefore, in this work, we have updated the tracking algorithm to utilize this new appearance model, so as to not require training on previously tracked objects. As stated previously, this now makes our tracking algorithm applicable to event types that do not have the previously tracked ground truth information available. We will describe this new model in Section 4.2.2, but first we will provide context to our problem of tracking multiple solar events by discussing the tracking problem that we are interested in solving. Then we will show how the online sparse coding appearance model is used for determining which detection is the most likely path taken by the tracked event being processed.

4.1 Tracking Multiple Objects in Video Data

The process of tracking multiple objects in video data, which is a difficult and only partially solved problem in computer vision, generally uses a process called multiple hypothesis tracking (MHT) to find an approximate solution. The tracking problem is generally broken into two independent steps that address separate issues of the overall goal of being able to track multiple interacting objects in video [50]. The first of these steps is the detection of targets in each frame of the video data, and is independent of any knowledge of time. In this work, it is assumed that the FFT software modules have accomplished this task and have reported the results to the HEK. The second step would then be to link detections over time into the most likely trajectories, based on some model of likelihood. This process is generally called the data association problem, as the

problem being solved is to associate one detection with its next detected appearance at a later time, and is what MHT is designed to solve.

In order to link detections into trajectories, MHT evaluates many alternative track formation hypotheses, which require a probabilistic expression of the data association problem. This probabilistic expression generally includes the prior probability of the presence of the target (object being tracked), how likely the detection is to be a false detection, and several different aspects of similarity between the potential detections in a particular path [1]. In our previous works of [17] and [18], this likelihood function incorporated similarity of trajectory movement and appearance, among other factors, with the goal of associating sensor reports into related groups that represent the paths of solar activity tracked as it crosses the observable solar disk.

4.1.1 Probabilistic Expression of Tracked Solar Activity

The input to the MHT algorithm is a set of detection reports $O = \{o_i, i \in \{1, \ldots, n\}\}$ from the HEK, for one specific event type, where each of the detection reports $o_i \in O$ is a set of attributes that uniquely describes a solar event in a particular location at a particular time. These attributes are

$$o_i = \begin{bmatrix} x_i : \text{spatial position} \\ s_i : \text{spatial outline (either minimum bounding rectangle or polygon outline)} \\ a_i : \text{physical appearance} \\ t_i : \text{time stamp} \end{bmatrix}$$

where x_i is the center of the bounding region of the detection, and s_i is the spatial outline of the detection. The physical appearance a_i is used to represent the appearance of the object using the image parameters from Table 2.1, which were derived from AIA images. Finally, t_i is the time stamp of the start time of the object detection so that the proper image parameters can be queried from the dataset.

Then, in MHT, an individual hypothesis of the path taken by a single object is defined as an ordered list of object detections, $T_i = \{o_{i_1}, o_{i_2}, \ldots, o_{i_n}\}$ where $o_{i_j} \in O$. This hypothesis is called a *trajectory or track hypothesis* and i_j of the individual observations o_{i_j} represents the j^{th} observation in hypothesis i. A global association hypothesis is then defined as a set of single trajectory hypotheses, $\tau = \{T_i\}$, given the current set of observations $O = \{o_i\}$ [51].

The data association problem, to which MHT is the solution, is represented as having the objective of maximizing the posterior probability of the global association hypothesis τ, given the current set of detections O. In doing so, it is showing that we wish to find the global association hypothesis τ that has the highest posterior probability from the set of all possible global association hypotheses $\tau^* = \{\tau_i\}$. The maximization problem is shown in Equation 4.1 and is called a maximum a posteriori (MAP) probability estimation problem. In this problem, we must assume that the likelihood probabilities are conditionally independent given the hypothesis τ, meaning that the knowledge about the truth of hypothesis τ does not change the belief in the likelihood of seeing observation o_i.

$$\begin{aligned} \tau^* &= \underset{\tau}{\operatorname{argmax}}\, P(\tau|O) \\ &= \underset{\tau}{\operatorname{argmax}}\, P(O|\tau)P(\tau) \\ &= \underset{\tau}{\operatorname{argmax}} \prod_i P(o_i|\tau)P(\tau) \end{aligned} \tag{4.1}$$

In order to meet this conditional independence requirement, the likelihood function of observation on o_i, denoted as $P(o_i|\tau)$, was formulated in [18] so that it does not rely on knowledge of the truth of the global association hypothesis τ, and thus τ does not change the belief in the likelihood of seeing observation o_i. In this context, $P(o_i|\tau)$ is trying to model how likely an individual detection report o_i is to be a false detection. This means that the detection algorithm of the FFT reported an observed object when

there was not one. Here, we make the assumption that a true target will generally persist across some number of detection report times.

To satisfy the conditional independence requirement, $P(o_i|\tau)$ is calculated as a Poisson probability of seeing the change in the number of detections from the previous frame to the current one, given the average change seen in the previous n frames. This is calculated as $f(\Delta;\lambda) = Pr(X = \Delta) = \frac{\lambda^{\Delta}e^{-\lambda}}{\Delta!}$, where Δ is the absolute change in the number of detections from the previous frame to the frame of o_i, and λ is the expected change, which is derived from the previous n frames. In order to simplify computation we want to avoid the calculation of $P(\Delta = 0; \lambda = 0)$. So, we chose to add one to all Δ and λ to account for these zero conditions. To correct the range of the likelihood of observation $P(o_i|\tau)$ for this zero condition fix, we then feed the value of $f(\Delta;\lambda)$ into a logistic function $l(x) = \frac{1}{1+e^{-k(x-x_0)}}$. As can be seen from the above formulas, each of these formulas is not affected by the knowledge of the truth of hypothesis τ, thereby matching the conditionally independent requirement for $P(o_i|\tau)$.

It should be noted that the optimization of Equation 4.1 is difficult due to the combinatorial growth in size of the set of association hypotheses in τ as the number of detection reporting times increases. In order to slow the growth of association hypotheses in τ, it is assumed that one observation o_i can belong to one and only one trajectory, or more formally defined as $T_i \cap T_j = \emptyset$, $\forall i \neq j$. Similarly, to simplify the calculation of $P(\tau)$, we assume that the motion of each object is independent so that Equation 4.1 can be rewritten as:

$$\tau^* = \underset{\tau}{\operatorname{argmax}} \prod_{o_i \in O} P(o_i|\tau) \prod_{T_i \in \tau} P(T_i) \qquad (4.2)$$
$$\text{Where } T_i \cap T_j = \emptyset, \forall i \neq j$$

Note that the prior of all trajectories in the global hypothesis τ has been broken down to the product of the priors of each trajectory $\prod_{T_i \in \tau} P(T_i)$. In the product, the prior probabil-

ity $P(T_i)$ of trajectory hypothesis T_i is modeled as a Markov chain of the n observations $[o_0, \ldots, o_n] \in T_i$. The chain includes an initialization probability $P_{entr}(o_0)$ at the detection of its initial time step, a termination probability $P_{exit}(o_n)$ at the detection of its final time step n, and the transition probabilities $P_{link}(o_{i+1}|o_i)$ between each detection in the interval between initial and final time steps:

$$\begin{aligned} P(T_i) &= P([o_0, o_1, \ldots, o_n]) \\ &= P_{entr}(o_0)P_{link}(o_1|o_0)P_{link}(o_2|o_1)\ldots P_{link}(o_n|o_{n-1})P_{exit}(o_n) \end{aligned} \tag{4.3}$$

A further discussion of the components of the Markov chain, and how they are used in finding the solution to the relaxed MAP probability estimation problem of Equation 4.2, will be presented in the next section.

4.2 Assigning Likelihood to Trajectory Hypotheses

In the previous section, the problem of multiple hypothesis tracking (MHT) was introduced as a maximum a posteriori (MAP) estimation problem of data association. In doing so, the data association problem defined the likelihood of each global hypothesis τ as the product of the prior probabilities of several trajectories $\prod_{T_k \in \tau} P(T_k)$. In the product, the prior probability $P(T_i)$ of trajectory hypothesis T_i was modeled as a Markov chain of the n observations $[o_0, \ldots, o_n] \in T_i$. The chain included an initialization probability $P_{entr}(o_0)$ at the detection of its initial time step, a termination probability $P_{exit}(o_n)$ at the detection of its final time step n, and the transition probabilities $P_{link}(o_{i+1}|o_i)$ between each detection:

$$\begin{aligned} P(T_i) &= P([o_0, o_1, \ldots, o_n]) \\ &= P_{entr}(o_0)P_{link}(o_1|o_0)P_{link}(o_2|o_1)\ldots P_{link}(o_n|o_{n-1})P_{exit}(o_n) \end{aligned} \tag{4.3 revisited}$$

In this section, we provide more details on the components of the Markov chain that represents the prior probability of a particular trajectory $P(T_k)$. We start with the initialization probability P_{entr}, as the cost of a hypothesis starting at a given track fragment, and the termination probability P_{exit} as the cost of a hypothesis ending at a specific track fragment. Then, we will discuss each of the components of the transition probability $P_{link}(o_{i+1}|o_i)$ for each track fragment in the hypothesis.

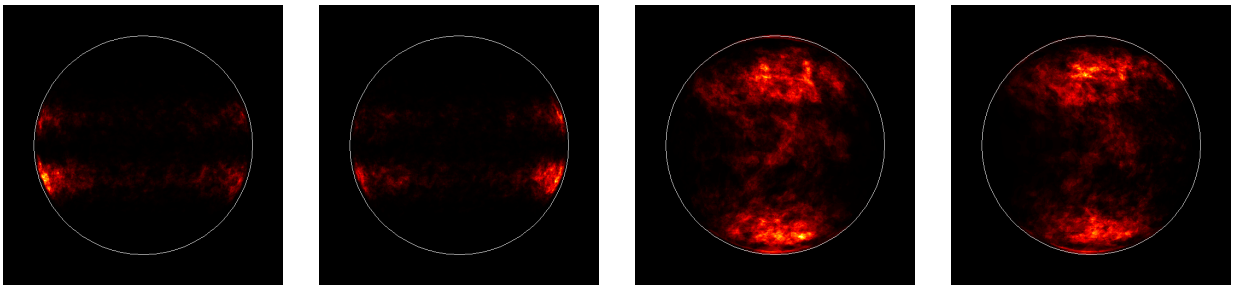
Here, we must note that this transition probability is formulated differently depending upon which step in our tracking algorithm is being performed. There are two variants of this formulation:

$$P_{link,variant_1}(o_{i+1}|o_i) = P(a_j|a_i)P(\Delta t|o_i) \tag{4.4a}$$

$$P_{link,variant_2}(o_{i+1}|o_i) = P(a_j|a_i)P(\Delta t|o_i)P(v_j|v_i) \tag{4.4b}$$

Our tracking algorithm attempts to find an optimal global hypothesis τ by performing several iterations of finding the solution to a smaller subproblem. The first iteration of this subproblem uses the first variant in Equation 4.4a, since all trajectory hypotheses are constructed from a single observation in this iteration. In subsequent iterations however, we use the second variant in Equation 4.4b, as there should now be multiple observations linked together in each trajectory hypothesis, which allows for movement comparison. In these variants, $P(a_j|a_i)$ is the appearance model, $P(\Delta t|o_i)$ the frame skip model, and $P(v_j|v_i)$ is the motion similarity model.

The following subsections will discuss each component of Equation 4.4. Section 4.2.1 will cover the initialization probability, P_{entr}, and termination probability, P_{exit}, in terms of one model. Then we will discuss the Appearance model $P(a_j|a_i)$ in Section 4.2.2, the Frame Skip Model $P(\Delta t|o_i)$ in Section 4.2.3, and finally the Motion model $P(v_j|v_i)$ in Section 4.2.4.



(a) AR Starting (b) AR Ending (c) CH Starting (d) CH Ending

Figure 4.2: Heat maps of event tracks' start and end locations, where the brighter colors indicate a higher percentage of starts/ends in that region. Where 4.2(a) shows where active regions tend to begin and 4.2(b) shows where active regions tend to end. Similarly, 4.2(c) shows where coronol holes tend to begin and 4.2(d) shows where coronal holes tend to end. The enter/exit model uses similar data to compute P_{entr} and P_{exit}.

4.2.1 Enter and Exit Probabilities

In the previous work done in [52], it was assumed that tracks can only start and end at the edges of the viewing area, with the exception of the starting and ending frames. This, however, does not hold true with the solar events in this work, since they may emerge or terminate as part of a natural, potentially chaotic process in many locations, with varying degrees of probability. We include Figure 4.2 as a visual aid to show where our tracking results have found tracks of two different types of solar activity and where they tend to start and end.

Since an event can start or end at almost any point of the solar surface in the viewing area, the enter and exit probabilities must be determined based upon the location and size of the event detections of interest. Note that the heat maps in Figs. 4.2(a) through 4.2(d) only show the location of trajectories starting and ending for a sample of our data (January 2012 and December 2014), but a similar pair of enter/exit maps was produced with many activity types all together and are used in this calculation. The maps used for enter/exit likelihood calculation were initially produced using the results from the track fragment forming algorithm that we use in feature selection for tracking (Algorithm 4.2), which will be discussed in Section 4.3.1. However, as results of the entire

tracking algorithm became available, we refined the maps with the output from the full implementation of our tracking work.

In order to calculate the likelihood of a trajectory starting or ending at any given pixel location, we use the following calculations. For P_{entr}, we let G be the global maximum probability on our enter probability map. Then let L be the local minimum probability on our enter probability map, or the pixel with the lowest probability inside the detection of interest. The calculation is the same for P_{exit}; the enter probability map is just substituted with the exit map. However, due to the fact that our heat map utilizes discrete pixels taken from a finite set of detections, there are areas in the constructed heat maps that contain a zero value when the probability of detection in those areas is greater than zero. In the cases that a detection's pixel intersects one of these areas, the average of all pixels in the detection of interest is assigned to L instead of the local minimum. The calculation of the enter $P_{entr}(o_i)$ probability is the ratio of L and G:

$$P_{entr}(o_i) = \frac{L}{G} \tag{4.5}$$

$P_{exit}(o_i)$ is calculated the same way using the exit heat map. Note, that when calculating $P_{entr}(o_i)$ of track fragment o_i, the first detection of the track fragment is used. Conversely, when calculating $P_{exit}(o_i)$ of track fragment o_i, the last detection in the track fragment is used. In either case, the probability of entering or exiting is always less than or equal to one.

4.2.2 Appearance Model

In the previous work done in [18], the appearance model $P(a_j|a_i)$ relied on learned global distributions of histogram similarity measures, which required previously tracked data in order to learn the expected distributions of these values. In [16], we developed an online sparse coding appearance model to eliminate this known ground truth requirement.

The use of sparse model representation as an instrument of single object visual tracking has seen promising results in the tracking research community [53–55]. It has also been used as a method for whole image classification and object recognition in regions of images, with [56, 57] being just a few examples. There has even been work in identification of anatomical structures in medical imaging [58, 59], using fuzzy images produced via an x-ray, which are similar to the raw AIA images from the SDO mission.

Recall from Section 2.2, that when using sparse models, it is assumed that a signal vector $x \in \Re^m$ can be represented as the linear combination of a small number of basis vectors. The representation uses basis vectors that come from a basis dictionary or codebook set with k entries $D \in \Re^{m \times k}$. Since the representation is generally not perfect, it is also assumed that some noise ϵ will need to be included as well:

$$x = D\alpha + \epsilon \qquad \text{(2.1 revisited)}$$

In this representation, vector α is assumed to have only a few significantly non-zero coefficients. To accomplish this assumption, an optimization problem with ℓ_1 regularization is utilized, since it is known that ℓ_1 regularization yields a sparse solution for α. By utilizing this regularization problem, the linear model $x \approx D\alpha$ is transformed to the quadratic program:

$$\underset{\alpha \in \Re^k}{\operatorname{argmin}} \frac{1}{2}\|x - D\alpha\|_2^2 + \lambda\|\alpha\|_1 \qquad \text{(2.2 revisited)}$$

where λ is a regularization parameter. The sparse coding problem in Equation 2.2 is a constrained version of the ordinary least squares regression problem known as basis pursuit [33], or the Lasso [34]. This ℓ_1 regularization problem is generally used because it yields a sparse solution for α. In our implementation, the optimal values of the coefficients of α in Equation 2.2 are found using the Least Angle Regression (LARS) algorithm

of [35]. By using the Lasso, we are looking for the "best matching" projection of the multidimensional input vector x onto our dictionary D, while keeping the solution sparse.

Constructing the Model

In this subsection, we discuss the various steps needed for our appearance model $P(a_j|a_i)$. For now, we assume that a set of image parameters has been chosen using the feature selection method to be described in Section 4.3.1, and we will create a set of input vectors for the appearance model using this set of chosen parameters. Recall from Section 2.2.1, that we construct a signal vector $x \in \Re^m$ from a sliding window of size p by p cells placed over a portion of the image being processed, where m is determined by the window size p and the number of image parameters we utilize. The image parameters from the p by p window are sequentially concatenated into a column vector. For example, using a window size of 4 by 4 cells (16 cell window), of 10 parameter values, over one filter channel, we would concatenate all of the values in this range (cell by cell) to form a vector of $m = 160$ values, which we denote as signal x.

This process is repeated by sliding the window by one cell and extracting the next signal from the current window area that partially overlaps the previous position. When using this sliding and extracting process, the set of all signals produced from the area of interest is denoted as signal matrix $\mathcal{X} = \{x_i | i = 1 : n\}$, where n is the number of windows that were extracted. The signal matrix is then later used to construct a sparse vector matrix A, which is the sparse representation of $\mathcal{X}$. Note that a matrix $\mathcal{X}$ is extracted for the target trajectory, or the trajectory we want to find a match for in a later time step, and each matching candidate trajectory also has a matrix $\hat{\mathcal{X}}$ extracted from it. The signal matrix from our target detection must also first be used for dictionary learning, as the resultant dictionary $D \in \Re^{m \times k}$ is used for construction of the sparse vector matrix A for both the target and candidate detections.

As was stated in Section 2.2.3, we have chosen to utilize the dictionary learning algorithm of [31], which can be found in Section 2.2.3 as Algorithm 2.2. The learned basis dictionary $D \in \Re^{m \times k}$, where k is the chosen number of basis vectors to learn, is a vital component of our appearance model. We learn a unique dictionary for each target trajectory being processed, where the target trajectory is the object that is projected forward to search for a matching candidate in subsequent time steps. This is done by first processing the bounding rectangle of the final detection in the trajectory, using the aforementioned extraction process to retrieve the signal matrix $\mathcal{X}$. Then, recall from Section 2.2.3, that given our signal matrix $\mathcal{X}$, the main objective of dictionary learning is to optimize some cost function:

$$f_n(D) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, D) \qquad \text{(2.6 revisited)}$$

where $D \in \Re^{m \times k}$ is the dictionary. Also recall that each column of D represents a basis vector, where ℓ is a loss function that decreases to some arbitrarily small value when the dictionary D represents the signal x in a sparse fashion, and with little error [31].

It should be noted that the number of samples n is determined by the size of the last detection in the target trajectory. Whereas, the signal dimension m is consistently set to 320 (4 cell by 4 cell windows with 20 parameters), and k is also constant, though determined by a user input when configuring the software. We have chosen $k < m$ to create a smaller dictionary dimension than our input dimension, effectively performing dimensionality reduction of the extracted, overlapping, 4 by 4 cell windows.

Once a dictionary is learned for the target detection, a coding histogram is then created using the coding histogram methods described below. For the vectors extracted from candidate detections, they are passed directly to the coding histogram creation. The results of the coding histogram creation and dictionary learning process are then combined to produce a candidate likelihood value in Section 4.2.2.

Coding Histogram

Recall that $\mathcal{X} = \{x_i | i = 1 : n\}$ of our target trajectory was produced by the sliding and extracting process, where n is the number of windows that were extracted, and the dictionary $D \in \Re^{m \times k}$ was learned from the extracted signals in $\mathcal{X}$. Now, with the dictionary $D \in \Re^{m \times k}$ learned, we can create our sparse representations α_i for each x_i in $\mathcal{X}$ of both our target trajectory and our candidate trajectories. In order to compare these sparse representations, we utilize the sparse coding histograms used by [53, 60] to represent the appearance distribution of the target and candidate models.

We begin by using the dictionary $D \in \Re^{m \times k}$, which is used to describe both the target (original) trajectory for which we wish to find a matching trajectory at a later time, and to describe the candidate (potential matching) trajectories. In the target model, we assume that $\mathcal{X}$ is the set of extracted signals from the last detection in the target trajectory. Whereas, in the candidate model, we assume that $\hat{\mathcal{X}}$ is the set of extracted signals from the first detection in the candidate trajectory. We then define $A \in \Re^{k \times n}$ as the set of n sparse vectors $\alpha \in \Re^k$ that satisfy the ℓ_1 regularization problem of Equation 2.2 for each $x_i \in \mathcal{X}$. We then similarly define $\hat{A} \in \Re^{k \times \hat{n}}$ as the set of $\hat{n}$ sparse vectors $\hat{\alpha} \in \Re^k$ that satisfy Equation 2.2 for each $\hat{x}_i \in \hat{\mathcal{X}}$. Finally, we define an isotropic Gaussian function:

$$g(\|c_i - \mu\|^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{\|c_i - \mu\|^2}{\sigma})^2} \tag{4.6}$$

where μ is considered the center of the area of interest, and $\|c_i - \mu\|^2$ is the square of the vector norm of the distance of c_i from the center of the area of interest. In the case of the target model, the area of interest is the minimum bounding box containing the last detection in the target trajectory, and is the minimum bounding box containing the first detection in the candidate trajectory for the candidate model. This is used to assign smaller weights to cells farther away from the center of the area of interest.

The target model is then constructed as the weighted bin values of the histogram constructed from A which are calculated as:

$$q_j = C \sum_{i=1}^{n} g(\|c_i - \mu\|^2)|\alpha_{ji}| \tag{4.7}$$

where C is a normalization constant to ensure $\sum_{j=1}^{k} q_j = 1$, and α_{ji} is the j^{th} coefficient of the i^{th} image patch. The candidate model histogram is similar to that of the target model histogram, in that the same isotropic Gaussian function $g(\cdot)$ is used to assign smaller weights to cell patches farther away from the center of the area of interest. The difference is that we use $\hat{A}$ as the set of $\hat{n}$ sparse vectors and compute the weighted bin values of the candidate histogram as:

$$\hat{p}_j = C \sum_{i=1}^{\hat{n}} g(\|\frac{\hat{c}_i - y}{h}\|^2)|\hat{\alpha}_{ji}| \tag{4.8}$$

where y is considered the center of the candidate area of interest, h is the scaling factor showing how much the candidate area of interest needs to be scaled to match the original target area of interest, and C is again the normalization constant used to ensure that $\sum_{j=1}^{k} \hat{p}_j = 1$.

Candidate Likelihood Evaluation

Recall that the appearance model $P(a_j|a_i)$ begins with our candidate trajectory having appearance a_i, and aims to determine the likelihood of said trajectory having appearance a_j at the time of the first detection in the candidate trajectory. With the target and candidate models produced from the coding histograms described earlier, and the sparse vector matrix $\hat{A} \in \Re^{k \times \hat{n}}$ used to create the candidate model, we want to come up with a final likelihood estimate for the appearance model. In doing so, we must evaluate the

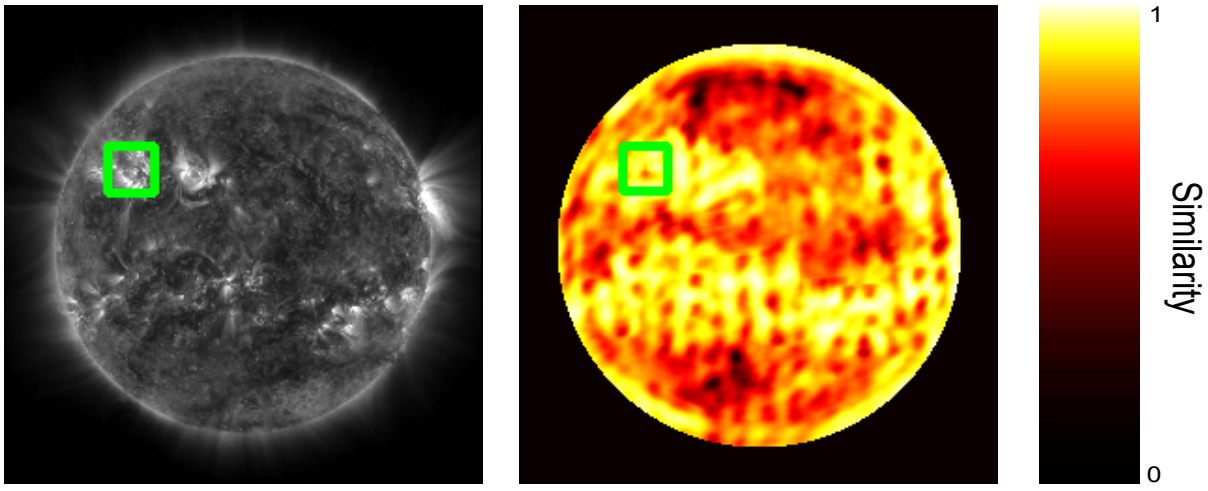
likelihood of the candidate detection being the same object that was used to train the dictionary (i.e., the target).

We begin by estimating the probability of a candidate detection centered at a given position being the target detection centered at that position in a later time. This is calculated as the generative likelihood of all the patches within the candidate window of interest:

$$P(y|D) = C\prod_{i=1}^{\hat{n}} e^{-g(\|\frac{\hat{c}_i - y}{h}\|^2)\frac{\epsilon_i^2}{\sigma^2}} \tag{4.9}$$

where ϵ_i is the reconstruction error for the i^{th} patch in the candidate area of interest, σ is the standard deviation of all the reconstruction errors for all the patches in the said area, and $g(\cdot)$ is still an isotropic Gaussian function as it was previously. Recall that each signal vector $\hat{x}$ of the candidate was approximated as $\hat{x} \approx D\hat{\alpha} + \epsilon$ using Equation 2.2, so the reconstruction is created as the product of the sparse vector representation $\hat{\alpha}$ and the learned dictionary D. Furthermore, in the generative likelihood calculation, we assume that y is the center of the candidate to be evaluated, while $\hat{X} = \hat{x}_i, i = 1 \ldots \hat{n}$ represents the $\hat{n}$ patches inside the candidate area of interest centered at y. As in [53], we ignore the constant C by taking the log of the likelihood for Equation 4.12.

Given that we are assuming that dictionary $D \in \Re^{m\times k}$ has been constructed from the extracted signals $X = \{x_i | i = 1 : n\}$ from the target trajectory, it is reasonable to expect that the local patches in the set of extracted signals X should have smaller reconstruction errors than the areas outside of the original window and so too should the most likely candidate. However, some regions that belong to the background can have a relatively large generative probability, since their appearances fall within the subspace spanned by the target dictionary basis. We include Figure 4.3 as an illustration showing these phenomenon. Here we use a bounding box to depict the original detection area that the dictionary was trained on (Figure 4.3(a)). Then, we plot the similarity values re-



(a) (b)

Figure 4.3: (a) The target appearance displays an active region in the 171Å filter channel. (b) The confidence map displays the 0–1 normalized likelihood values returned from the evaluation of the generative likelihood calculation in Equation 4.9 (described in Section 4.2.2) for each position on the observed solar surface at the 171Å filter channel. The brighter colors are seen as being more likely to be the area that was trained on based strictly on the reconstruction error of the area within a window placed at that point, when using the trained dictionary.

turned by the generative likelihood calculation for each position on the original image in Figure 4.3(b). In this figure, it is clear that strictly relying on reconstruction error is insufficient for discriminating between highly similar areas, as many areas can be reconstructed with little error when using the learned dictionary.

The problem of having some regions that belong to the background producing a relatively large generative probability can arise because of the introduction of a few contaminated bases. In some cases, these contaminated bases in the trained dictionary can be introduced by input into the training process that is selected from areas outside the actual object of interest. Though undesirable, it is difficult to avoid due to the imprecision of using the minimum bounding rectangle to encapsulate the object of interest and the use of square windows to extract a signal vector. Another reason for background regions producing large generative probabilities is that many regions on the solar surface have significant visual similarities.

This problem means that we cannot simply rely on how well we can recreate signals from the candidate region using the learned dictionary for our likelihood estimation. So, an additional step is added in order to obtain information about how similar the distribution of parameter values are in both the target and candidate regions. We begin by using the two coding histogram models described in Section 4.2.2, one that represents the target trajectory, and another that represents the candidate trajectory. These two coding histograms are then compared by using the Bhattacharyya metric to produce a similarity value for the sparse coding histograms of the target (q) and candidate ($\hat{p}$):

$$d(y) = \sqrt{1 - p(\hat{p}, q)} \tag{4.10}$$

$$p(\hat{p}, q) = \sum_{j=1}^{k} \sqrt{\hat{p}_j q_j} \tag{4.11}$$

In [53], the target is located in the image by maximizing:

$$\hat{p}(y|D) = p(\hat{p}, q)L(y|D) \tag{4.12}$$

where the $L(y|D)$ component is the log of Equation 4.9 and measures the probability of the candidate being generated from the target dictionary D, and the $p(\hat{p}, q)$ term is from the Bhattacharyya metric $d(y)$, which weights the probability value in proportion to how well the distribution of the target model and the candidate model match.

Unlike [53], we only evaluate the set of already detected target candidates, and are not using Equation 4.12 to search the entire image for a least negative value. Therefore,

we modify the equation to produce values that we can interpret as the likelihood of the target and candidate matching. To do this, we utilize the logistic function:

$$\hat{p}(y|D) = \frac{1}{1 + \exp^{-k \times ((p(\hat{p},q)L(y|D)) - m_0)}} \tag{4.13}$$

where k is the steepness variable and m_0 is the expected midpoint of the possible values from Equation 4.12. This provides the $[0, 1]$ range that we require as an output from the appearance model. Thus, the output of the appearance model would be $P(a_j|a_i) = \hat{p}(y|D)$ where D is learned from a_i, and the output shows the likelihood of a_j being the next detection of a_i.

4.2.3 Frame Skip Model

The frame skip model $P(\Delta t|o_i)$, is used to account for detections that are not reported in the dataset, but the tracked object still exists. This is a common issue as no object detection algorithm can guarantee perfect identification in every image. This leads to the need to allow for missed detections of an object and to continue tracking an object across these gaps.

In the model, the Δt is the gap between the last detection of the target trajectory and the first detection of the candidate trajectory. It should be noted that each detection is considered to have a period of time over which it is valid and another detection is not expected to be reported to the dataset. For example, active regions have a reporting cadence of roughly every 4 hours. Therefore, the first skipped frame would be after 8 hours, the second after 12, and so on. This period is different for each event type in the dataset.

With this information in mind, we define the frame skip model as follows:

$$P(\Delta t|o_i) = \prod_{t \in F} (1 - P_{exit}(o_i)) \tag{4.14}$$

where F is the set of skipped frames in Δt. As was previously stated, one time period is allowed from the start of the last object detection report o_i of the target trajectory, and is considered the valid period of that detection. At the end of that valid period, a search is conducted for spatiotemporally neighboring candidate trajectories that do not have a skipped frame. A search is then conducted for candidate trajectories at the time of the valid period plus one skipped time period or frame. The search is repeated up to the maximum of n skipped frames, where n is dependent upon the limit set at each iteration of the tracking algorithm. Note that the maximum number of skipped frames n is increased at each iteration of the tracking algorithm, thereby giving priority to candidates with fewer skipped frames.

4.2.4 Motion Model

In the second variant of the transition probability calculation, Equation 4.4b, there is one more term $P(v_j|v_i)$, which considers the motion similarity of the two compared trajectories. This motion model, similar to [52], uses the normalized movement vectors of the candidate track fragment o_j and the target track fragment o_i. Specifically, the normalized mean of the frame-wise movement within the trajectories is calculated. The means are labeled as v_j for the candidate trajectory and v_i for the target trajectory. The model value is then calculated as:

$$P(v_j|v_i) = 1 - \frac{1}{2}\|v_j - v_i\| \tag{4.15}$$

This comparison of the Euclidean norm of movement differences assumes that the motions of the detected objects are not changing direction abruptly, which is a fair assumption for the solar events with which we are concerned. The result of this comparison is the probability of the two trajectories being the same object based solely upon how similar their movement is.

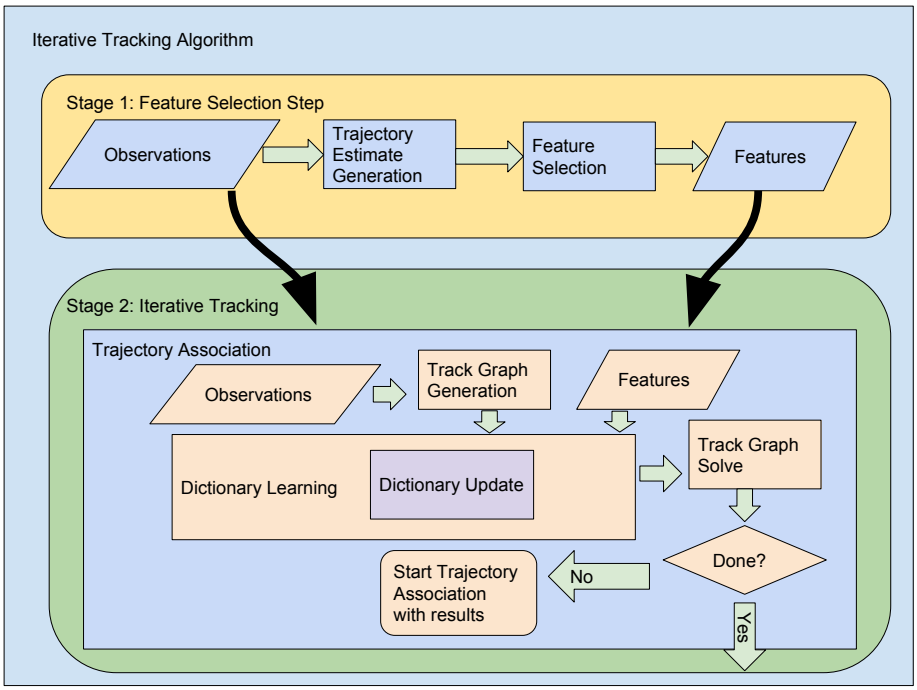


Figure 4.4: Tracking algorithm flow diagram.

4.3 Using Iterative Refinement for the Global Hypothesis Solution

In previous sections, we described multiple hypothesis tracking (MHT) as having an objective of maximizing the probability of a global association hypothesis τ. This problem was called a maximum a posteriori (MAP) probability estimation problem and was formulated as:

$$\begin{aligned}\tau^* &= \underset{\tau}{\operatorname{argmax}}\, P(\tau|O) \\ &= \underset{\tau}{\operatorname{argmax}}\, P(O|\tau)P(\tau) \qquad \text{(4.1 revisited)} \\ &= \underset{\tau}{\operatorname{argmax}} \prod_i P(o_i|\tau)P(\tau)\end{aligned}$$

Which was then simplified, through various independence assumptions, to be:

$$\tau^* = \underset{\tau}{\operatorname{argmax}} \prod_{o_i \in O} P(o_i|\tau) \prod_{T_i \in \tau} P(T_i) \qquad \text{(4.2 revisited)}$$
$$\text{Where } T_i \cap T_j = \emptyset, \forall i \neq j$$

In this, the global hypothesis τ was broken down to the product of the priors of each trajectory $\prod_{T_i \in \tau} P(T_i)$. These prior probabilities $P(T_i)$ of trajectory hypotheses T_i were each modeled as a Markov chain of the n observations $[o_0, \ldots, o_n] \in T_i$. These chains included an initialization probability $P_{entr}(o_0)$ at the detection of its initial time step, a termination probability $P_{exit}(o_n)$ at the detection of its final time step n, and the transition probabilities $P_{link}(o_{i+1}|o_i)$ between each detection in the interval between initial and final time steps:

$$\begin{aligned} P(T_i) &= P([o_0, o_1, \ldots, o_n]) \\ &= P_{entr}(o_0) P_{link}(o_1|o_0) P_{link}(o_2|o_1) \ldots P_{link}(o_n|o_{n-1}) P_{exit}(o_n) \end{aligned} \qquad \text{(4.3 revisited)}$$

Then, the transition probability $P_{link}(o_{i+1}|o_i)$ was shown to contain multiple components and be formulated differently depending upon which step in our tracking algorithm is being performed:

$$P_{link,var_1}(o_{i+1}|o_i) = P(a_j|a_i) P(\Delta t|o_i) \qquad \text{(4.4a revisited)}$$
$$P_{link,var_2}(o_{i+1}|o_i) = P(a_j|a_i) P(\Delta t|o_i) P(v_j|v_i) \qquad \text{(4.4b revisited)}$$

where $P(a_j|a_i)$ was the appearance model, $P(\Delta t|o_i)$ was the frame skip model, and $P(v_j|v_i)$ was the motion similarity model.

In this section, we discuss how we approximate the solution to the global MAP estimation problem through iteratively solving the MAP problem on a constrained subset of the input needed for the global solution. In Algorithm 4.1, this is accomplished in the

Algorithm 4.1 Iterative Tracking Algorithm

Input: O (The set of observation detections)
Input: maxSkip (The maximum allowed number of skipped frames between detections)
Output: tl (list of tracked object hypotheses)

1: **function** IterativeTracking(O, maxSkip)
2: tl ← ∅ ▷ An empty doubly linked list of track hypotheses
3: skip ← 0 ▷ The allowed frame skip in current iteration
4: feat ← ∅ ▷ Empty set of image features (parameters)
5: ▷ Stage 1
6: tl ← TrjEstGen(O) ▷ Create a list of track hypotheses using Algo. 4.2
7: feat ← FeatSelect(tl) ▷ Get list of features for this event type using Algo. 4.3
8: tl ← ∅ ▷ Clear list of track hypotheses
9: $\forall o_i \in O : tl \leftarrow T_i$ ▷ Set each observation as a track hypothesis to process
10: ▷ Stage 2
11: **while** skip < maxSkip **do**
12: tl ← ASSOC(tl, feat, skip) ▷ Link track hypotheses using Algo. 4.5
13: skip = skip + 1 ▷ Increase allowed skipped frames for linking
14: **end while**
15: **return** tl
16: **end function**

Iterative Tracking step or Stage 2, using the Trajectory Association Algorithm 4.5 repeatedly. The flow diagram of the overall tracking procedure of Algorithm 4.1 is depicted in Figure 4.4. As can be seen in Algorithm 4.1, the tracking algorithm starts with the set of observations of a particular event type, and then uses them in the FeatSelect Step of Stage 1 to find the top-rated features from Table 2.1 in the various wavebands of the image data. Then, the resultant set of top-rated features and the original set of detections are passed into stage two, where the individual detections are linked into tracks over several iterations of Iterative Tracking step in Stage 2, using the Trajectory Association Algorithm 4.5 repeatedly. How many iterations of Algorithm 4.5 are used is dependent upon the user defined setting of maxSkip.

Below, we break each stage into its own section. We discuss the Feature Selection of Stage 1 in Section 4.3.1. This feature selection process uses the Top-K feature selection method discussed in Chapter 3, as it was found to work as well as more complicated methods and is a more efficient process than the others that were evaluated. Then,

in Section 4.3.2, we begin to explain how the MAP problem is solved using multiple iterations of Algorithm 4.5, which is further broken down into a data association process solved in Section 4.3.3.

4.3.1 Tracking Stage 1: Feature Selection

As shown in Figure 4.4 and Algorithm 4.1 above, the first step in the tracking process is to do feature selection on the image parameters. In this step, we wish to find a set of relevant image parameters from various wavebands which will provide a good basis of visual comparison for individual detections in the actual tracking portion of our algorithm. Feature selection is a critical step that is used to select a subset of the input for use in classification and is performed as a pre-processing step in our tracking algorithm. The feature selection process executes on the image parameters from each waveband in our image dataset. It ranks all the features by ordering them in descending value of their calculated F-Statistic value. The top K are selected as the feature set to use for the solar event type being processed.

It should be noted that the previous discussion on feature selection in Chapter 3 assumed that tracks had already been formed with several reports in each track. This was the case with the data used in the feature selection evaluations, but it can no longer be assumed with the tracking algorithm, as the main motivation of tracking is to produce these tracked objects and because the feature selection process is performed as a prerequisite of tracking. This means that we need to produce an initial approximation of tracked events in order to complete the feature selection process. To create the approximation of tracked events, we utilize an heuristic method to estimate the tracked objects. In this heuristic algorithm, track estimates are created by linking object detection reports that are highly likely to belong to the same track.

The process of making a set of track estimates is done in Algorithm 4.2 where spatiotemporal neighbors are searched for using differential rotation of the sun to project

Algorithm 4.2 Trajectory Estimate Generation Algorithm

Input: O (set of observations)
Output: tl (list of tracked object hypotheses)

```
1:  function TrjEstGen(O)
2:      tl ← An empty list of trajectory hypotheses (T_i)
3:      (doubly linked lists)
4:      for each detection o_i ∈ O do in parallel
5:          l_1 ← Forward_ST_Neighbors of o_i
6:          l_2 ← Back_ST_Neighbors of o_i
7:          if l_1 count is 1 and l_2 count ⩽ 1 then
8:              o_j ← Forward_ST_Neighbor of o_i
9:              if o_j previous is NULL then
10:                 o_j.previous ← o_i
11:                 o_i.next ← o_j
12:                 tl ← T_i = o_i ∪ o_j
13:             else
14:                 tl ← T_i = o_i
15:             end if
16:         else
17:             tl ← T_i = o_i
18:         end if
19:     end for
20:     Clear duplicate trajectory hypotheses from tl
21:     return tl
22: end function
```

where to search. Each detection in the set of observations O is processed in parallel, and only those that have a single spatiotemporal neighbor, both going forward and backward in the time dimension, shall be linked together as a trajectory. By using only spatiotemporal neighbors, who only have the current detection as a spatiotemporal neighbor, we can allow certain data dependency issues to be ignored; by doing so, this algorithm can take significant advantage of parallel execution. For instance, due to the filtering constraints to determine whether two detections shall be linked, no detection will be selected as the next in the track by more than one previous detection.

This means that if two detection reports get added to different tracks by two different threads of execution, these track fragments will be equivalent by the constraint that only one spatiotemporal neighbor was available for each detection to be linked with.

Therefore, any duplicate results produced by parallel execution can simply be pruned and discarded at the conclusion of the grouping process. Once all observations $o_i \in O$ are processed, the duplicate trajectories are pruned from the results and the result set tl is returned. These results are then passed on to Algorithm 4.3 for the actual feature selection process.

The feature selection process then finds the top-rated image parameters from the set of parameters listed in Table 2.1 in the various wavebands of the image data. It does this by utilizing the set of trajectory hypotheses tl from the trajectory estimate generation algorithm to compare the histograms of image parameter values produced by detections within the same trajectory hypothesis and those from different trajectory hypotheses. In order to compare two event reports and get a feature value for a particular parameter, a 20-bin histogram of cell values within the bounding box of an event report is extracted. This process is repeated for both reports being processed. These histograms are then compared using a common histogram comparison method and the resultant value of the comparison is used as the feature value.

The histogram comparison method in this case is the histogram intersection, and is defined as the sum of the minimum bin count between the two histograms being compared or:

$$d(H_1, H_2) = \sum_{I}^{N} min(H_1(I), H_2(I)) \tag{4.17}$$

where $H_k(I)$ is the value of the histogram bin at index I of N bins in histogram k. This comparison method was chosen as an approximation of the amount of mutual information between the two histograms and was found to most effectively generalize for ranking after trying several different comparison methods in [44]. The feature selection algorithm (Algorithm 4.3) then uses the histogram intersection feature values and calculates the ANOVA F-Statistic using the two populations.

Algorithm 4.3 Feature Selection

Input: tl Set of trajectory hypotheses
Output: Pl_{fl} (list of parameters to use in tracking)

```
1:  function FeatSelect(tl)
2:      Pl ← The list of all image parameters
3:      fl ← List of F-stat values ∀p_i ∈ Pl
4:      for all p_i ∈ Pl do
5:          dl ← Empty list of distance pairs
6:          for each track hypothesis t_i ∈ tl do in parallel
7:              if t_i.length > 1 then
8:                  for all detection pairs d_i ∈ t_i do
9:                      dl ← pair of histogram distances for
10:                     (1) pair d_i
11:                     (2) first detection in d_i and
12:                     random detection not in t_i
13:                 end for
14:             end if
15:         end for
16:         fl ← F-stat for p_i
17:     end forreturn Pl_fl          ▷ Top−K parameters from Pl ranked on F-stat in fl
18: end function
```

In order to create the two populations of feature values, we only utilize those trajectories that contain more than one detection. So, each temporally sequential pair in these trajectories is used as a comparison point belonging to the class labeled as "same object." Then, the first detection of each of these pairs is taken and compared to another detection from the data set that resides in another trajectory. These comparisons are considered as being in the class labeled as "different object." This process results in a balanced 50/50 split between the two classes. Once the two classes of measurements are complete for a particular image parameter on a particular AIA channel, the F-Statistic is calculated and the value is stored for use once all parameters are evaluated. This process repeats for all 90 AIA channel and image parameter pair combinations and then all of the features are ranked by ordering them in descending value of their calculated F-Statistic value. The Top$-$K ($K = 20$) are then returned as the feature set to use for the solar event type being processed, which was deemed sufficient in Chapter 3. The track estimates are broken

down after the feature selection algorithm is finished, and the original set of observations is passed on to Stage 2 with the selected feature list from Algorithm 4.3.

4.3.2 Tracking Stage 2: Iterative Tracking

As can be seen in Figure 4.4, Stage 2 is given an input of the individual observation reports and the feature set selected by Stage 1. The goal of this stage is to find the most likely trajectories from the input set of observations, which is not a trivial task. In tracking single objects in the set of observations, object trajectories arise by simply connecting the detections of the single object over time. The addition of multiple detections in any given time step breaks this naïve approach and the likelihood of each path must then be considered. In addition, failures in detection also become an issue, such as false positive detections or detections that were missed.

To solve the additional complexity that is found in tracking multiple objects at once, the problem becomes how to associate multiple detections into multiple trajectories across time. The data association approach of multiple hypothesis tracking is used because it can be quite robust to these occasional failures in detection. For example, in the case of a false positive, erroneous detections are often isolated in time and are therefore easily discarded. Similarly, recovery from failed detections can be incorporated to produce an uninterrupted trajectory [50].

In order to recover from failed detections, our implementation allows "skipped frames", or time steps that erroneously do not have a detection report for a given trajectory. This is accomplished by searching for spatiotemporal neighbors of the trajectory currently being processed in a timestamp that is later than what would be expected for the next observation in the same object trajectory. However, this increases the number of trajectory hypotheses that need to be evaluated, as we then need to consider all the possible trajectories in each of the intervening time steps.

To illustrate this, let us assume that an object detection report o_{k_i} (k_i denotes the i^{th} frame and the k^{th} object report in said frame) can be associated with any object detection report in a subsequent frame $o_{k_{i+1}}$. Then, when searching m frames, with n object reports per frame, the number of all possible track paths is bound by $\theta(n^m)$. This means that finding the optimal global trajectory hypothesis τ is an optimization problem that grows in complexity at an exponential rate with the number of reporting frames [61]. To help mitigate this exponential growth problem, we adopt a similar approach to that used in [52]. In this approach, there is a hierarchy of stages in the tracking algorithm that iteratively grows track fragments as they are passed from one stage to the next. This set of stages reduces the complexity of finding the global optimal trajectory by making the assumption that detections should be linked to temporally closer detections before more temporally distant solutions are even considered.

This assumption first limits the exponent m by creating trajectory fragments that have no gaps or "skipped frames" in them. This is accomplished by only searching the next frame for spatiotemporal neighbors. By doing this, the number of object detection reports available for possible association in subsequent frames is limited as a result of the independence assumption mentioned in Section 4.1. This assumption states that detections can belong to one and only one trajectory. So, once a detection belongs to a trajectory, it cannot be considered for association in a subsequent stage, and thus reduces n for stages that allow m to grow.

In Figure 4.4, we have changed the naming slightly from [52] in that the processing of results from previous solutions are simply called iterations of Stage 2. In the first iteration of Stage 2, the input is all single detections, which are treated as trajectory fragments of length one. After the first iteration of Stage 2, most of the trajectory fragments are longer than a single detection. As such, this iteratively growing framework uses the simpler transition probability $P_{link}(o_{i+1}|o_i) = P_{link,var_1}$ from Equation 4.4 on the first iteration where a majority of the object detection reports will be linked into larger trajec-

tories. Then, the more complex transition probability P_{link,var_2} is used in later iterations where most trajectories are of a length that previous motion can be used for comparison.

We utilize these two variants of the transition probability calculation for two reasons: (1.) Because it is not possible to compare movement similarity when trajectory fragments are length one in the first iteration. (2.) We do not want to exclude movement similarity altogether, because greater ambiguity is present in the association hypotheses in later iterations due to larger number of skipped frames allowed, and more information about the similarity is helpful. It should be noted that the input observations to the transition probability $P_{link}(o_{i+1}|o_i)$ are trajectory fragments and are treated as individual observations in our notation.

In order to search for a possible next observation to link to, and to calculate the transition probability, a search area must first be created. However, as stated previously, these trajectory fragments are only one detection in length in the first iteration of Stage 2. So, the search area is created by using the differential rotation of the sun from [62] to estimate how far a detection might move in the elapsed time before the next detection report, and the bounding box of the detection is projected by that estimate. After the first iteration of Stage 2, there are now trajectory fragments longer than a single detection. So, it is now possible to use the previous motion of the trajectory fragments to predict where the next trajectory fragment may be located. In the cases where trajectory fragments are now sufficiently long enough, meaning there are more than two reports, the previous motion is used to predict the next trajectory fragment location. Also, the subsequent iterations of Stage 2 allow the search for candidate track fragments to occur more than one time period in the future, which is an attempt to allow for missed detections.

Once the spatiotemporal neighbors are found for all of the trajectory fragments in the current dataset, and the transition probabilities are calculated for those neighbors, the most probable trajectory paths need to be found. This is called the data association problem, and is solved as the maximization of the *a posteriori* probability. This association

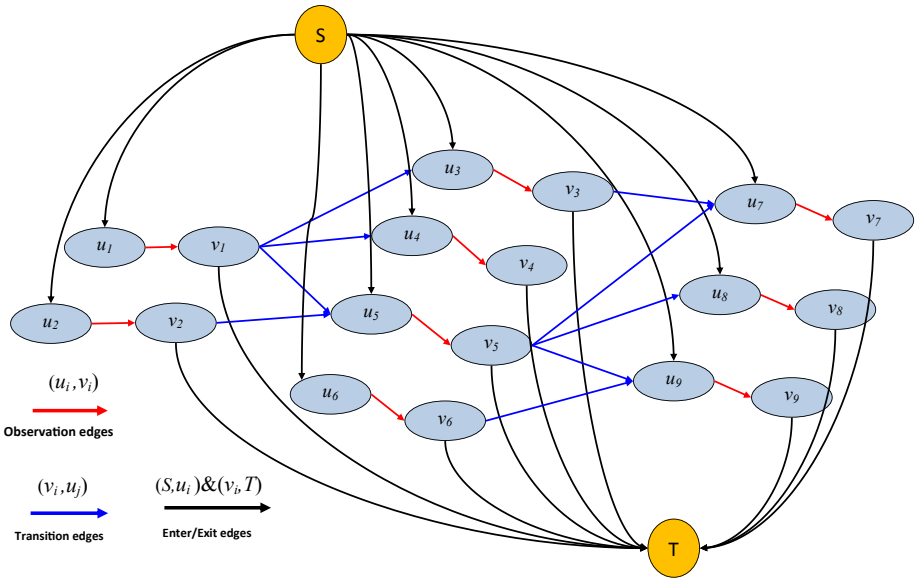


Figure 4.5: An example of the cost-flow network used to associate detections into longer track fragments. This graph depicts a set of detections with three time steps and nine observations, see Section 4.3.3 for a detailed description of the components of this graph.

is done at each iteration of Stage 2, slowly building an approximation of the global maximization of the a posteriori probability by using the resultant trajectories from the previous iteration as the starting dataset for the next. In the next section, we describe how this association process is accomplished through the use of a minimum cost-flow problem on a directed acyclic graph.

4.3.3 Data Association Through Track Graphs

In the previous section, we discussed how the tracking framework shown in Figure 4.4 must solve the data association problem at each iteration. In this section, we begin our discussion of the association problem and utilize graphs as a means to solve it. We convert the association problem to a minimum cost-flow problem in a directed acyclic graph. In doing so, we convert the MAP probability problem, presented in Equation 4.1, to find the minimum cost flow through a graph. We call this graph the track fragment graph, where the vertices are associated with trajectory fragments; the fragments may

be individual detections of an object or multiple detections linked together in previous steps to form a longer trajectory fragment.

In the track graph, a hypothesis of an object's movement is a path in the graph, and a global association hypothesis is a set of consistent trajectories. In order to maintain consistency, no two trajectories in the global association hypothesis can share a single object detection report o_i. Edges/links that connect two vertices are created if and only if the nodes have the possibility of representing the same object trajectory at differing timestamps. To be considered as a possible representation of the same object trajectory, the start time of the successor trajectory fragment must be later than the end time of the predecessor trajectory fragment.

As was mentioned in Section 4.3.2, if it is assumed that any object detection report o_{k_i} (k_i denotes the i^{th} frame and the k^{th} object report in said frame) can be associated with any object detection report in a subsequent frame $o_{k_{i+1}}$, then for m frames with n object reports per frame, the number of all possible trajectory paths is $\theta(n^m)$. This means that the optimal hypothesis is found by solving an optimization problem, which grows exponentially in size with the number of frames [61], and explicitly storing each of these track paths will also grow exponentially. So, we take two approaches to help mitigate this growth in storage requirements. The first of which was discussed in Section 4.3.2, where we iteratively lengthen the allowed skipped frames as trajectories get longer and trajectory fragments get fewer. The second is the graphical formulation of the trajectory association problem in which hypotheses are implicitly represented as paths through the graph, instead of having to store each path explicitly.

The graphical formulation approach to mitigating storage growth will still have the exponential rate of growth in the number of trajectory hypotheses as dictated by the data when the number of potential matches increases. However, it accomplishes this with increasing the number of nodes and edges in the graph at a linear rate with respect to the number of trajectories represented in the data. It is because of this linear rate

Algorithm 4.4 Track Graph Construction

Input: $\mathtt{tl}$ (is a list of track hypotheses)
Output: G (graph constructed from $\mathtt{tl}$)

```
1: function CONSTRUCT(tl)
2:     G ← graph with source S and sink T
3:     for each track hypothesis T_i ∈ tl do
4:         G ← vertices u_i, v_i
5:         G ← edge (u_i, v_i) weight C_i                ▷ Equation 4.18d
6:         G ← edge (S, u_i) weight C_{en,i}             ▷ Equation 4.18a
7:         G ← edge (v_i, T) weight C_{ex,i}             ▷ Equation 4.18b
8:     end for
9:     return G
10: end function
```

of growth that the graphical representation was chosen for our work over an explicit construction of every possible trajectory hypothesis.

Track Fragment Graph

In Figure 4.5 we show a depiction of the track fragment graph G. In it, the track fragment graph G is initialized with a source vertex S and a sink vertex T. This is the first step shown in Algorithm 4.4 and is not dependent upon the data. Then, given the set of trajectory fragments O, labeled as track list $\mathtt{tl}$ in Algorithm 4.4, we initialize the graph with two vertices u_i and v_i for each $o_i \in O$, or for each track hypothesis. As each pair of vertices u_i and v_i is added, an edge (u_i, v_i) is inserted between each of the u_i and v_i pairs that were inserted for each of the $o_i \in O$. Then an edge from the source vertex S is inserted into the graph to every vertex u_i, and for each of the vertices v_i there will be an edge going to the terminating vertex T. Finally, for those trajectory fragments determined to be spatiotemporal neighbors in Algorithm 4.5 by the search criteria described in Section 4.3.2, an edge is added from the initial trajectory fragment o_i to each of the candidates o_j. The edges are created from vertex v_i of searching trajectory fragment o_i to vertex u_j of neighboring candidate trajectory fragment o_j. The example track fragment graph in in Figure 4.5 has nine such trajectory fragments being represented.

Algorithm 4.5 Trajectory Association

Input: tl (Set of trajectory hypotheses)
Input: feat (List of image parameters to use)
Input: skip (Current allowable skipped frames)
Output: tl (Updated list of track hypotheses)

function Assoc(tl, feat, skip)
    G ← Conscruct(tl) ▷ Algo. 4.4

    for each track hypothesis $t_i \in tl$ **do in parallel**
        l_1 ← Forward_ST_Neighbors of t_i
        allowing for skip number of skipped frames

        for all $t_j \in l_1$ **do**
            if skip < 1 **then**
                G ← edge $e_{i,j}$ from t_i to t_j
                $e_{i,j}$ ← weight from $P_{link,variant_1}$
            else
                G ← edge $e_{i,j}$ from t_i to t_j
                $e_{i,j}$ ← weight from $P_{link,variant_2}$
            end if
        end for
    end for

    Link track hypotheses $T_i \in tl$
    using results from Solve(G) ▷ Algo. 4.6

    Remove duplicate track hypotheses $T_i \in tl$
    return tl
end function

Next, we turn our attention to the weights of the edges we just defined. The weights, or costs of traversing an edge in the graph, are calculated with the use of the Markov chain parameters that were discussed previously in Section 4.2; the initialization probability

$P_{entr}(o_i)$, the termination probability $P_{exit}(o_i)$, and the transition probability $P_{link}(o_j|o_i)$:

$$C_{en,i} = -\log P_{entr}(o_i) \tag{4.18a}$$

$$C_{ex,i} = -\log P_{exit}(o_i) \tag{4.18b}$$

$$C_{i,j} = -\log P_{link}(o_j|o_i) \tag{4.18c}$$

$$C_i = \log \frac{1-\beta_i}{\beta_i} \tag{4.18d}$$

In these calculations, we get the cost of a trajectory starting at a given trajectory fragment $C_{en,i}$ in Equation 4.18a, the cost of a trajectory ending at a given trajectory fragment $C_{ex,i}$ in Equation 4.18b, and the cost of two trajectory fragments being joined into one larger trajectory fragment $C_{i,j}$ in Equation 4.18c. The final cost in Equation 4.18d contains β, which is used to model the cases of an observation being a true detection or a false detection $\beta_i = 1 - f(\Delta;\lambda)$, where $f(\Delta;\lambda)$ was discussed in Section 4.1.1 as the Poisson probability of seeing the change in detections from one frame to the next.

Association by Solving the Track Fragment Graph

In Algorithm 4.5, after the track fragment graph G is constructed, the data association problem can be solved as a minimum cost-flow problem in G. In order to show how the track graph is used to solve the data association problem, we begin by defining the

following $0-1$ indicator variables that are used to indicate hypothesis membership for a particular trajectory, as was done in [51,61,63]:

$$f_{en,i} = \begin{cases} 1 & \exists \tau_k \in \tau, \tau_k \text{ starts from } o_i \\ 0 & \text{otherwise} \end{cases} \tag{4.19a}$$

$$f_{ex,i} = \begin{cases} 1 & \exists \tau_k \in \tau, \tau_k \text{ ends at } o_i \\ 0 & \text{otherwise} \end{cases} \tag{4.19b}$$

$$f_{i,j} = \begin{cases} 1 & \exists \tau_k \in \tau, o_j \text{ immediately follows } o_i \text{ in } \tau_k \\ 0 & \text{otherwise} \end{cases} \tag{4.19c}$$

$$f_i = \begin{cases} 1 & \exists \tau_k \in \tau, o_i \in \tau_k \\ 0 & \text{otherwise} \end{cases} \tag{4.19d}$$

Here, as previously mentioned, o_i is considered a trajectory fragment, with potentially many object detection reports. Each of the the indicator variables is one when o_i is in trajectory hypothesis τ_k of the global hypothesis τ, and zero otherwise. The various indicators are as follows:

- $f_{en,i}$ is the flow into the vertex u_i from the source vertex, which means the beginning of the trajectory hypothesis is at trajectory fragment o_i referenced by this vertex
- $f_{ex,i}$ is the flow out of the vertex v_i to the termination vertex, which indicates the end of the trajectory hypothesis is at trajectory fragment o_i referenced by this vertex
- $f_{i,j}$ is the flow from vertex v_i, representing trajectory fragment o_i, to vertex u_j, representing trajectory fragment o_j, which means that the two trajectories o_i and o_j belong to the same trajectory hypothesis

- f_i indicates the referenced trajectory fragment o_i, represented by vertices u_i and v_i, belonging to the hypothesis of interest τ_k

Recall that there is a constraint that trajectory hypotheses are non-overlapping. So, in order for the global trajectory hypothesis set τ to be consistent, the following constraint must be met:

$$f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \tag{4.20}$$

This simply states that the flow into a vertex of the graph is the same as the flow out of that same vertex, which is the same as the flow between any two vertices in the graph that correspond to the same trajectory hypothesis τ_k. By satisfying the constraints set forth in Equation 4.20, the flow through any given path in the graph is limited to one, which means that no trajectory or part of a trajectory can participate in more than one trajectory hypothesis.

Now that the indicator variables and the constraint using those variables have been defined, we can move on to converting the MAP probability into a minimization of the cost of sending a flow through the network of graph G. To convert the maximization

problem indicated in Equation 4.1 into the minimum-cost flow variant, the negative log of the posterior is taken as follows:

$$\begin{aligned}
\tau^* &= \underset{\tau}{\operatorname{argmin}} \sum_i -\log\, P(o_i|\tau) + \sum_{\tau_k \in \tau} -\log\, P(\tau_k) \\
&= \underset{\tau}{\operatorname{argmin}} \sum_i (-\log\, (\beta_i) f_i - \log\, (1-\beta_i)(1-f_i)) \\
&\quad + \sum_{\tau_k \in \tau} (C_{en,k_0} f_{en,k_0} \\
&\quad + \sum_j (C_{k_j,k_{j+1}} f_{k_j,k_{j+1}}) + C_{ex,k_n} f_{ex,k_n}) \\
&= \underset{\tau}{\operatorname{argmin}} \sum_i (C_{en,i} f_{en,i}) + \sum ij (C_{i,j} f_{i,j}) \\
&\quad + \sum_i (C_{ex,i} f_{ex,i}) + \sum_i (C_i f_i)
\end{aligned} \tag{4.21}$$

The final iteration of Equation 4.21 is subject to the flow constraints set out in Equation 4.20, and the cost of flow through the vertices is given by the functions defined in Equations 4.18a-4.18d.

Note that the construction of the observation weight in Equation 4.18d gives a negative cost to edges associated with trajectory fragments that have an observation likelihood value greater than 0.50. This negative cost allows the optimal cost of traversing the edges in G to become negative by sending flow through these negative-cost edges. The negative weighting produced by Equation 4.18d is needed because of the simple fact that if all the edge costs in G were positive, the minimum-cost flow would be the trivial empty zero-cost flow [51], and would result in no linking of smaller track hypotheses into larger track hypotheses.

The flow through the edges in G are that of the indicator variable in Equations 4.19a-4.19d, where the flow is 1 when the track hypothesis traverses the edge, and will be 0 otherwise. For each new flow through the graph, a new association hypothesis is formed. Recall that edges were created from searching target trajectory fragment o_i to neighbor-

Algorithm 4.6 Track Graph Minimum-Cost Flow Solve

Input: $G(tl)$ a graph constructed from list tl of trajectory hypotheses T_i
Output: tl_1 (list of hypothesis paths that decrease the cost of flow through $G(tl)$)

```
1:  function SOLVE(G(tl))
2:      τ_i is a hypothesis path in G(tl)
3:      while ∃ τ_i ∈ G(tl) with negative cost do
4:          Find minimum-cost path τ_i from S to T
5:          not already used.
6:          if Cost τ_i is negative then
7:              tl_1 ← τ_i
8:              τ_i ← flow in G(tl) as 1 (used)
9:          else
10:             Break
11:         end if
12:     end while
13:     return tl_1
14: end function
```

ing candidate trajectory fragment o_j, and the cost of traversing one of these edges would be the cost of associating these two trajectory fragments into a new trajectory fragment that encompasses both. In [63], it was shown that the cost of the optimal solution in this directed acyclic graph is convex and can be solved by solving $K+1$ shortest-path problems, where K is the optimal number of trajectory hypotheses.

Each iteration increases the flow through the graph by 1 and decreases the cost of flow by the cost of hypothesis path τ_i. For a graph with an optimal number of trajectory fragment association hypotheses of K, the algorithm terminates after $K+1$ iterations, having found a minimum cost flow where any further flow from S to T would only serve to increase the cost of flow through the network [63]. The algorithm for this process of finding the optimal K trajectory association hypotheses is presented as Algorithm 4.6. Once the algorithm terminates, the trajectory fragments that are part of the same hypothesis path through the graph are linked by taking the last detection in target trajectory fragment o_i and linking it to the first detection in candidate trajectory fragment o_j.

This process produces duplicate results in the same fashion as those returned from the track estimate creation heuristic in the feature selection process described in Section 4.3.1,

where there are two trajectory fragment pointers pointing to the same linked list. As was described in Section 4.3.1, the duplicate pointers are pruned to only include one per linked list. The result set after pruning is returned as the set of current trajectory fragments, which are either passed onto the next iteration or returned as the final results dependent upon the iterative stopping criteria.

5 IMAGE DESCRIPTORS FOR SDO AIA IMAGES

Recall from Section 2.4, that in order to aid the exploration of the massive image archive produced by the SDO mission, a provisional Content-Based Image Retrieval (CBIR) system was developed for retrieval of similar whole images within the dateset [64]. While preparing for the CBIR system, [22–24] found that texture-based image parameters work well for image comparison in this domain. In [39], the most similar images were found to be from the preceding and succeeding time steps, and a few possible solutions for this phenomena were explored. One possible solution was to increase the time between samplings of the dataset. However, as was noted by [39], this leads to some events in the data being completely missed, as their lifespan is shorter than the sampling period. Ultimately, the similarity of temporal neighbors problem was left as an open issue to be addressed in future work.

Later, in our work on tracking solar events in [16], we presented a method based on sparse coding that was capable of differentiating between different classes of the same solar event type. In that work, we showed that the use of sparse coding was more selective than histograms of image parameters in this same solar event type differentiation task. Given its capability to differentiate between regions of such similar visual characteristics in AIA images, we postulated that sparse coding should also be capable of differentiating between similar whole images as well. The use of sparse model representation is a popular method for whole image classification and object recognition in regions of images, with [56, 65] being just a few examples. There has been work in identification of anatomical structures in medical images [58, 59], which utilizes fuzzy images produced in the X-ray wavelength and that have similar characteristics to our raw AIA images from the SDO mission. In [65], sparse coding was used to extract salient properties of

appearance descriptors from images of differing types. These properties have also been pooled at multiple spatial scales and local features have been computed from each sub-region [56, 65, 66]. However, to our knowledge, the use of sparse coding has never been applied to AIA images.

5.1 Whole Image Vector Descriptor

Our first investigations in using sparse coding as a means of producing image descriptors began with creating sparse image descriptors for whole SDO AIA image retrieval in [40]. In the whole image descriptor work, we introduce an image descriptor that reduces the dimensionality of the SDO AIA images even more than previously produced parameters of [23] listed in Table 2.1. In doing so, the reduced dimensionality descriptors can have indexing techniques applied to them in order to speed up retrieval.

In the production of a descriptor for an image, we begin by extracting a signal vector that is representative of a specific location in an image. This extraction process is similar to the one described previously in Section 2.2.1 in that the signal vector $x \in \Re^m$ is from a sliding window of size p by p cells placed over a portion of the region of interest being processed. Just like the previous extraction process, this is done by concatenating the values from each cell onto the vector produced by those values in the previous cell. For example, using a window size of 4 by 4 cells, the 16 cells of 10 parameter values over one wavelength, would be concatenated together to form a vector of $m = 160$ values, which we denote as signal x. However, in the case of producing a descriptor for an image of a particular wavelength, the entire image is our region of interest, and the cell values from all 10 image parameters within this window are placed into the column vector.

In order to use the extracted signal vectors, we must first learn a dictionary to use as a learned set of basis vectors $D \in \Re^{m \times k}$, where k is the chosen number of basis vectors to learn. Just like tracking, we utilize the dictionary learning algorithm presented in Section 2.2.3, but unlike tracking, we only learn a unique dictionary for each of the

9 wavelengths utilized from our dataset. The learned dictionary for a particular wavelength is then used as the dictionary for the corresponding wavelength of images in order to create our descriptor for a given image. Each of these dictionaries is learned using many sample images taken from the dataset. Each one of the sample images has the extraction process applied to it, and all of the resultant signal vectors are placed into a large signal vector matrix, which we denote as X. Again, this is done independently for each of the 9 wavelengths we utilize.

So, as was described in Section 2.2.3, given our signal matrix X, constructed from the sample images, the main objective of dictionary learning is to optimize a cost function

$$f_n(D) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, D) \qquad \text{(2.6 revisited)}$$

where D in $\Re^{m \times k}$ is the dictionary. Each column of D represents a basis vector, where ℓ is a loss function that decreases to some arbitrarily small value when the dictionary D represents the signal x in a sparse fashion, and with little error [31]. It should be noted that the number of samples n is determined by the number of images we choose to use for dictionary learning. Whereas, the signal dimension m is consistently set to 640 (8 by 8 cell matrix with 10 parameters), and k is also constant at 256 so that $k \ll n$. We have chosen $k < m$ to create a smaller dictionary dimension than our input dimension, effectively performing dimensionality reduction on each of the extracted, overlapping, 8 by 8 cell windows.

After the dictionary D is learned, the next step is to extract a set of signal vectors $x_i \in \Re^m$ for an image and produce the signal matrix X for the image being processed. Then, each of the signal vectors x_i, constructed from the sliding window extraction process, need to be represented as a sparse vector α_i. In this approximation, it is assumed that

each signal vector $x_i \in \Re^m$ can be represented as a linear combination of a small number of the basis vectors in our dictionary with k entries $D \in \Re^{m \times k}$ and a noise factor ϵ:

$$x_i = D\alpha_i + \epsilon \qquad \text{(2.1 revisited)}$$

As was described in Section 2.2.2, the optimal values of the coefficients of α_i in Equation 2.1 are found using the Least Angle Regression algorithm of [35] for the ℓ_1-regularlized Lasso problem in Equation 2.7. Recall that, by using the Lasso, we search for the "best matching" projection of the multidimensional input vector x_i onto our dictionary D, while keeping the solution sparse.

At this point, each of the signal vectors $x_i \in \mathcal{X}$ are represented as a weighted sum of the elements in D, by the sparse vectors $\alpha_i \in \mathcal{A}$ where $\mathcal{A} \in \Re^{k \times n}$. This resultant matrix of vectors is then used to compute an image feature vector similar to that done in [56] and [65]. The feature vector is created by computing $\mathbf{z}$ for each image, using a pre-chosen pooling function $\mathbf{z} = \mathcal{F}(\mathcal{A})$ where $\mathcal{F}$ is defined on each row of $\mathcal{A}$.

5.1.1 *Max Pooling*

Specifically, we use a spatial pooling technique similar to that used in [65], which was originally proposed in [66]. In the pooling method, each of the rows of $\mathcal{A}$ correspond to the responses of all the signal vectors $x_i \in \mathcal{X}$, to a specific item (column) in the dictionary D. We chose to use the max pooling function for $\mathcal{F}$ on the absolute value of the sparse codes in each row of $\mathcal{A}$

$$z_i = max\{|\alpha_{i,1}|, |\alpha_{i,2}|, \ldots, |\alpha_{i,n}|\} \qquad (5.1)$$

where each z_j is the j-th element of $\mathbf{z}$, and each $\alpha_{i,j}$ is the coefficient in the i-th row and j-th column of the sparse feature matrix $\mathcal{A}$.

Our choice of max pooling does have some biological reasoning, in that, the output response of the max pooling method would be dominated by the best match of any part of the input stimulus to the dictionary elements. This mechanism is similar to when multiple stimuli are in the receptive field of a neuron and its response is dominated by the stimulus that, when presented in isolation, produces the higher response rate [67].

5.2 Image Region Vector Descriptor

After our first use of sparse coding for whole AIA image retrieval in [40], and our use of sparse coding for region comparison for tracking in [16], our next step is to produce a method for region comparison for retrieval. In doing so, we still wish to have the same constraints as were outlined for whole AIA image retrieval, specifically that we want a descriptor that is small enough to index, i.e., less than 150 dimensions.

Just as done before with whole image descriptors for retrieval and for region descriptors for tracking, we begin by extracting a signal vector that is representative of a specific location in an image. This extraction process is similar to the one described previously in Sections 2.2.1 and 5.1, in that the signal vector $x \in \Re^m$ is from a sliding window of size p by p cells placed over a portion of the region of interest being processed. Unlike in Section 5.1 though, for our region descriptors, we are again considering regions of interest that are part of the image and not the entire image. Also unlike Section 5.1, instead of extraction only the 10 image parameters of a single wavelength, we are utilizing all 90 available parameters (10 parameters $\times$ 9 wavelengths). In doing so, each signal vector $x \in \Re^m$, produced by the extraction process, shall be 360 dimension with a $p = 4$ sized window.

In order to use the extracted signal vectors, we must first learn a dictionary to use as a learned set of basis vectors $D \in \Re^{m \times k}$, where k is the chosen number of basis vectors to learn. We utilize the dictionary learning algorithm presented in Section 2.2.3, but unlike either of the two previous applications, we learn a global dictionary from the input set

of 360 dimensional signal vectors $x \in \Re^m$ that were extracted from all wavelengths and all the image parameters. The global dictionary is learned from sample signal vectors produced from applying the extraction process to a number of sample regions of interest. The resultant signal vectors were placed into a large signal vector matrix, which we label as $\mathcal{X}$.

So, again, as was described in Section 2.2.3, given our signal matrix $\mathcal{X}$, constructed from the sample images, the main objective of dictionary learning is to optimize some cost function

$$f_n(D) \triangleq \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, D) \qquad \text{(2.6 revisited)}$$

where D in $\Re^{m \times k}$ is the dictionary. Each column of D represents a basis vector, where ℓ is some loss function that decreases to some arbitrarily small value when the dictionary D represents the signal x in a sparse fashion, and with little error [31]. The number of samples n is determined by the number of regions of interest we choose to learn from along with their size. The signal dimension m is consistent, though it set to 360 (4 by 4 cell matrix with 90 parameters) in our region descriptor process. Finally, the number of dictionary elements k is constant for a given set of experiments, but we did investigate a number of values for k to find what setting works for our application. The actual value of k will be discussed more in Section 6.3, but all values are well below the 150 dimensions we set forth as a limit in the opening of this section.

After the dictionary D is learned, the next step is to extract a set of signal vectors $x_i \in \Re^m$ for an image and produce the signal matrix $\mathcal{X}$ for the image being processed. Then, each of the signal vectors x_i, constructed from the sliding window extraction process, need to be represented as a sparse vector α_i. In this approximation, it is assumed that

each signal vector $x_i \in \Re^m$ can be represented as a linear combination of a small number of the basis vectors in our dictionary with k entries $D \in \Re^{m \times k}$ and a noise factor ϵ:

$$x_i = D\alpha_i + \epsilon \qquad \text{(2.1 revisited)}$$

Again, just as was done before, and described in Section 2.2.2, the optimal values of the coefficients of α_i are found using the Least Angle Regression algorithm of [35] for the ℓ_1- regularlized Lasso problem in Equation 2.7. At this point, each of the signal vectors $x_i \in \mathcal{X}$ are represented as a weighted sum of the elements in D, by the sparse vectors $\alpha_i \in \mathcal{A}$ where $\mathcal{A} \in \Re^{k \times n}$. This resultant matrix of vectors is then used to compute the region feature vector similar to that done for tracking in [16]. Just as was done for the whole image descriptors, the region feature vector is created by computing $\mathbf{z}$ for each image, using a pre-chosen pooling function $\mathbf{z} = \mathcal{F}(\mathcal{A})$ where $\mathcal{F}$ is defined on each row of $\mathcal{A}$. The function $\mathcal{F}$ is described in the following subsection.

5.2.1 *Histogram of Sparse Codes*

As mentioned in Section 5.2, a matrix of sparse vectors $\mathcal{A}$ was constructed for a region of an image, however, we want a single vector to describe the entire region. In [16], we were able to do this using a histogram of the sparse weights. However, this histogram was of the absolute value of the weights, and didn't treat a positive weight any different than a negative. In our descriptors for region retrieval, we found that this should also be taken into account. So, our function $\mathcal{F}$ shall now do so for region descriptors and is described as Algorithm 5.1. In Algorithm 5.1, as was done for tracking and [16], the final step is to scale the histogram vector to have unit length. It is this normalized histogram vector that is used for our region descriptor $\mathbf{z}$ in our experiments described in Chapter 6.

Algorithm 5.1 Calculates a Sparse Histogram from $\mathcal{A}$

Input: $\mathcal{A}$ The set of observations
Output: $\mathtt{hist}$ (histogram of normalized coefficient weights)

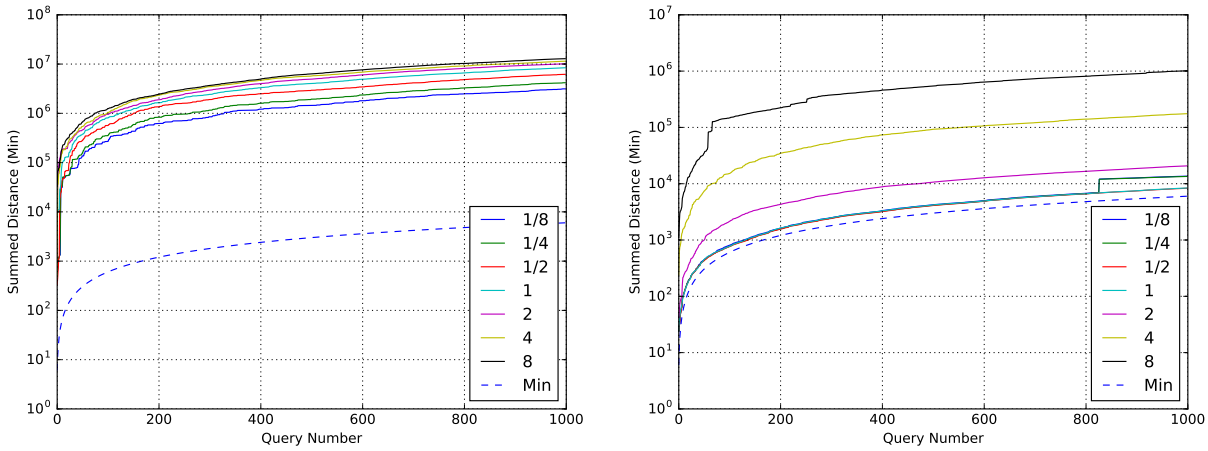
function CalcSparseHisto($\mathcal{A}$)
  $\mathtt{hist} \leftarrow$ Array of length $2 * k$
  for i from 1 to k **do**
    for j from 1 to n **do**
      $\alpha_{ij} \leftarrow \mathcal{A}_{ij}$ ▷ Value of $\mathcal{A}$ at Row i Column j
      if $\alpha_{ij} < 0$ **then**
        $\mathtt{hist}_{2*i} \leftarrow \mathtt{hist}_{2*i} + |\alpha_{ij}|$
      else
        $\mathtt{hist}_i \leftarrow \mathtt{hist}_i + \alpha_{ij}$
      end if
    end for
  end for
  $\mathtt{hist} \leftarrow \frac{\mathtt{hist}}{\|\mathtt{hist}\|}$ ▷ Scale to unit length
  return $\mathtt{hist}$
end function

5.3 Choosing the Norm for Image Comparison

With the assumption that an image descriptor **z** has been created, we need to determine a method of comparison for the descriptors, so that we can find the images in our dataset that are similar to one another. In content-based retrieval systems, use the distance between the user query input and each element in the dataset to retrieve the most similar [68]. However, in high-dimensional space, all pairwise distances can be very similar. This problem, often referred to as the curse of dimensionality, also affects the original image parameter data in our dataset, as was found in [39], where a large number of temporal neighbors all have very similar distances from each other.

In both [68] and [69], this phenomenon was discussed in the context of using Minkowski norms (also called L_p norms). These norms are a family of metrics that are parameterized by their exponent $1 \leqslant p \leqslant \infty$: For a $x_j = [x_{j1}, \ldots, x_{jd}] \in \mathfrak{R}^d$

$$\|x_j\|_p = \Big(\sum_i |x_{ji}|^p \Big)^{\frac{1}{p}} \tag{5.2}$$



(a) (b)

Figure 5.1: Summed temporal distance (in minutes) for first nearest neighbor over 1K queries, 5.1(a) for image parameters, and 5.1(b) for whole image sparse descriptors.

In addition, the fractional norms were also discussed, where $0 < p < 1$, and it was concluded that the "optimal" value of p is highly application dependent. Given this fact, to determine the p−norm that is most suitable for the given application, it was proposed that using a statistical measure of relevance for each p−norm considered would be an elegant solution to choosing the proper norm. So, as done in [68], we tested several p−norms, using a relevance feedback strategy to pick the the most applicable for our data.

In our relevance feedback strategy, we consider the most similar image in the dataset to be, on average, the immediate temporal neighbor of a query image. So, for each of the distance measures, we start by randomly choosing 1,000 query images from the month of January 2012, and scan the entire month to find the one image that is the most similar to the query. Using the result, we then calculate how many minutes away the returned most similar image is from its query image, and sum the distance in time over the 1,000 trials.

In Figure 5.1(a), the results of this feedback strategy, on the pooled image descriptors, is displayed for for 131Å wavelength images. It can be seen that the $p = \frac{1}{8}$ fractional norm seems to provide the most relevant results according to our assumptions, as it is

the closest to the minimum value achievable for this test. In Figure 5.1(b), this test is repeated using the original image parameter space so that we can compare our results to that of [39]. These results are similar to what was found by [39], in that we see very little difference between the fractional norms and the $p = 1$ (Manhattan distance) norm. This information, coupled with the general requirement of indexing techniques that a distance function must satisfy the triangle inequality property, leads us to choose the fractional norm with $p = 1$ as the default distance measure for the rest our work.

6 EXPERIMENTAL EVALUATIONS

In this chapter, we will discuss the experiments that were conducted on the various use cases presented in Chapters 4 and 5. We begin with discussing the evaluation of our tracking use case from Chapter 4 in Section 6.1. Then we present our evaluations of our whole image and region descriptors from Chapter 5, starting with the whole image descriptors in Section 6.2 and conclude with region descriptors in Section 6.3.

6.1 Tracking Evaluation

In this section, we evaluate our algorithm on solar data retrieved from [27] and [70]. As in [17] and [18], we will be utilizing active region and coronal hole detections for evaluation. Unlike most other solar phenomena reported to the HEK, these event types contain tracking data from the original Spatial Possibilistic Clustering Algorithm (SPoCA) detection module [46]. We have also curated a dataset of human labels for the ground truth information on two months of these active region and coronal hole reports, so as to have a basis of comparison for our method against the original labels.

We will begin by describing the human labeling process in Section 6.1.1 as well as how we utilized the produced data to create a ground truth. We then provide a brief description of the evaluation metrics we use for comparing the human-labeled dataset to our results and the SPoCA module results to the human-labeled dataset in Section 6.1.2. Then, in Section 6.1.3, we discuss the tuning of weighting constants that are used to emphasize various components of the trajectory hypothesis when calculating the edge weights in the track graph G. Finally, in Section 6.1.4, we provide the comparison of our algorithm and the SPoCA module results to the human-labeled dataset.

6.1.1 *Crowd Sourced Human Labels*

For our crowd sourcing of human labels, we utilized non-expert human labelers to determine the sequence of detections representing a tracked object. To accomplish this, we presented the labelers with an original detection, chosen at random, from either our active region or coronal hole dataset. In an effort to reduce any bias that could be introduced from not specifically accounting for changing behavior of solar events as the solar cycle progresses, we used detections from two distinct months, January 2012 or December 2014. These months were chosen because they were the most distant months within our current image parameter dataset (2012/01/01 00:00:00–2014/12/31 23:59:59 UTC). Given their temporal distance, these time frames would be more likely to exhibit independent behavior of the tracked objects due to their differing points in the solar cycle.

With the original detection presented to the labeler, we then present all of the spatiotemporal neighbors in the next n timesteps as potential matches for being the next detection in the sequence. The labeler can choose to see a video sequence of what this will look like prior to making their choice. The labelers are allowed to choose a sequence of up to three detections at a time to label as being consecutive steps in a trajectory of a tracked object.

Since we are using non-expert human labelers, we opt to not rely on one person's opinion on what the next step of the sequence should be. Instead, we use the majority voting strategy, which counts each labeler's decision as one vote. In Figure 6.1, we show that every original detection in our two-month dataset has at least three votes. There are approximately 8,600 votes on 1,561 coronal hole detections, and 14,100 votes on 2,647 active region detections. The median number of votes on any particular detection is five in both cases.

Then, in Figure 6.2(a), we show the percentage of labels going to each of the choices presented to the labelers as possible next steps in the sequence. For example, if there

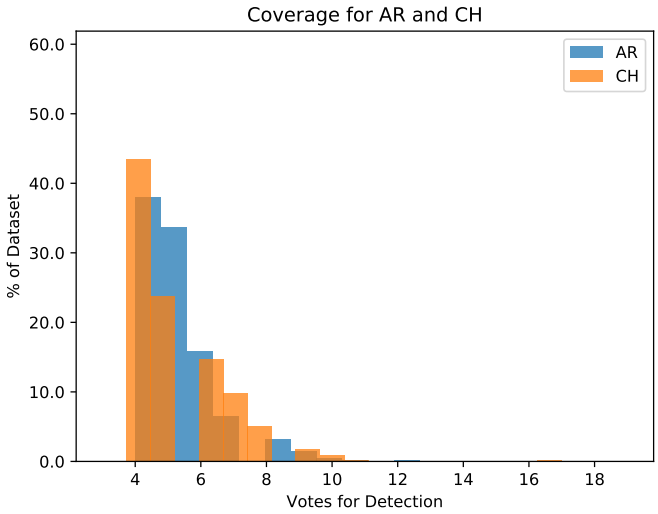
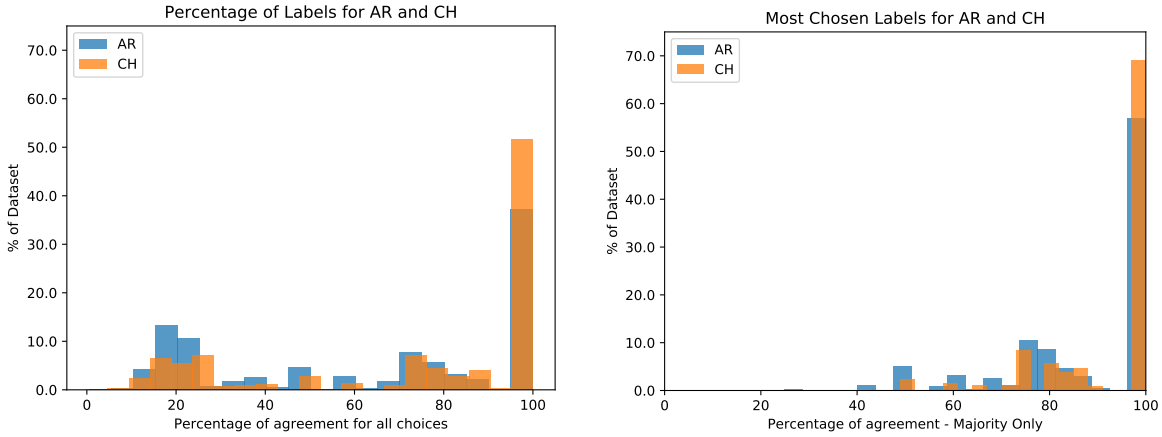


Figure 6.1: Histogram showing the distribution of how many times each detection in our dataset was used as the initial detection in a vote for a sequence.



(a) (b)

Figure 6.2: Plots showing the percentage of the user labels that point to the same successor for a given detection. (a) Distribution of the percentage of labels for all successors pointed to by a given predecessor. (b) Distribution of the percentage of labels for the most chosen successor for each detection.

are two choices of detections for the next step in a sequence, and every labeler chose the same detection, this would mean that 100% of the labels go to the same choice. In the figure, we can see that this 100% agreement of which detection is next in the sequence happens for about 70% of the coronal holes and 60% of the active regions.

However, since Figure 6.2(a) includes all choices made by the labelers, it is difficult to see how much of the dataset has a plurality of votes going to one choice and how much has an absolute majority going to one choice. So, to make this easier to discern, in Figure 6.2(b) we show only the majority votes in our dataset. In other words, for each detection in the dataset that has a successor, we show the percentage of votes that went to the successor with the most votes. In it, we again see this 70% and 60% range of our dataset with a high confidence in the user decisions.

We also see in Figure 6.2(b) that almost every set of user decisions gives us greater than an absolute majority of user votes going to the same next detection. Though having any part of the dataset below a 100% agreement of votes is not ideal, we believe that 66% is an acceptable level of ambiguity, as this guarantees that at least 2 of 3 votes have been for the same next detection in the sequence.

6.1.2 Evaluation Metrics

The metrics used to evaluate the accuracy of the tracking algorithm are listed in Table 6.1. The definition of *Mostly Tracked (MT)* is from [71], which is when the ground truth trajectory is covered by one output trajectory from the tracking algorithm for greater than 80% of its length. Note that we assume a trajectory is mostly tracked when 80% of the total frames are covered by the ground truth trajectory, regardless of identity switches.

The definition of *(Identity) Switches* is when a tracked object from the tracking output is following the path of a particular tracked object in the ground truth and it then switches to another. As an example, if the output indicates that a person wearing a green shirt

Table 6.1: Definitions of Tracking Evaluation Metrics; MOTA, MT, and Switches are from [71] and [72].

Name	Definition
MOTA	Multiple Object Tracking Accuracy: $1 - E_{tot}$ Where E_{tot} is the ratio of the sum of all errors (misses, false positive, mismatches) and the total number of objects in each frame. Larger is better. Range $[0\%, 100\%]$
MT%	Mostly Tracked: Percentage of the ground truth trajectories that are covered by one track output for more than 80% of its length. Larger is better. Range $[0\%, 100\%]$
Switches	Identity Switches: The average number of times the tracking output of one tracked object switches from one ground truth track to another. Smaller is better. Range $[0, n]$ where n is the number of detections in the dataset. Though the theoretical upper limit is n, a more practical upper limit would be the number of tracks that occur over the lifetime of the ground truth track.
Track Count	The total number of tracks in the results of the tracking algorithm being applied.
Avg. Length	The average number of detections associated into one track in the results of the tracking algorithm being applied.
Max Length	The largest number of detections associated into one track in the results of the tracking algorithm being applied.

and walking along from one frame to the next is the identity of the track, and then at some frame it switches over to the person wearing a red shirt that was walking beside them, this would be considered an identity switch of the tracking output. Then, if the tracking output switched back to the person in the green shirt, this would be registered as another identity switch. The Identity Switch metric is simply the average number of times this occurs over all ground truth trajectories in our data.

The definition of the metric *Multiple Object Tracking Accuracy (MOTA)* from [72] provides another description for the accuracy of the tracking algorithm. The MOTA metric sums all the errors at each time step (misses, false positives, mismatched trajectory labels) and then computes the ratio of errors to the total number of objects over all time steps. This process gives the total error rate E_{tot}, and $1 - E_{tot}$ is the resulting MOTA metric value.

6.1.3 *Weighting Constants*

As described in Section 4.3.3, we build the track graph G by utilizing Equation 4.18a-4.18d to provide weights to the edges. However, we found that weighting some of the edges more heavily than others produces better results than a naïve approach of simply using the raw weights from each of the equations. To that end, we conducted a search for an improved weighting scheme by allowing a weighting constant to vary on each of the equations and compare the results of tracking to our human-labeled ground truth to find the best results.

We conducted this search by having a single constant for Equations 4.18a and 4.18b called the *Enter/Exit Multiplier Constant*, a constant for Equation 4.18c called the *Transition Multiplier Constant*, and a constant for Equation 4.18d called the *Observation Multiplier Constant*. We started each of these multipliers at 5 and searched the three dimensions allowing Enter/Exit to vary up to 45, Observation up to 205, and Transition up to 195. The best results we found were around Enter/Exit = 5, Observation = 65, and Transition = 175.

6.1.4 *Tracking Performance*

As was mentioned in Section 6.1.1, our human-labeled data utilized active region and coronal hole detection from two months, January 2012 and December 2014. We use the results from the human labeling as the ground truth with which to compare our tracking results. We also compare the SPoCA module results to the same set of human-labeled data. The results of the comparison of both our tracking and the SPoCA tracking are listed in Table 6.2. In it, we see that our tracking algorithm outperforms the SPoCA module for active regions. However, we also see that our tracking module does slightly worse than the SPoCA module for coronal holes.

Table 6.2: Results when comparing SPoCA and our tracking method to human-labeled ground truth tracks. Results are shown for both unlimited solar radius ($R_\odot$), and limited to within 80% of the $R_\odot$. The evaluation metrics that show improvement when limiting the area in which tracking is done are in bold.

Tracking Results using Human-Labeled Ground Truth								
Data			**MOTA**	**MT%**	**Switches**	**Track Count**	**Avg. Length**	**Max Length**
AR	**SPoCA**	**100% $R_\odot$**	0.902	0.662	0.681	485	5.45	73
		80% $R_\odot$	0.876	0.643	0.736	323	4.18	44
	Track	**100% $R_\odot$**	0.923	0.675	0.541	399	6.63	75
		80% $R_\odot$	**0.925**	**0.736**	**0.445**	221	6.11	44
CH	**SPoCA**	**100% $R_\odot$**	0.976	0.889	0.156	130	12.01	108
		80% $R_\odot$	0.986	0.842	0.193	58	13.88	46
	Track	**100% $R_\odot$**	0.969	0.877	0.197	220	7.1	108
		80% $R_\odot$	**0.991**	**0.895**	**0.123**	49	16.43	46

To see why our tracking module was not performing as well as the SPoCA module on coronal holes, we looked at the cases where our tracking algorithm disagrees with the human labels for both active regions and coronal holes in Figure 6.3. Here we see that many of our mislabeling happens at the limbs of the solar disk, mostly in the greater than 80% of solar radius ($R_\odot$) area between the inner and outer circles. According to [8], some of the reporting modules limit detection to only a percentage of the solar disk, so we also investigated limiting our tracking to only being applied to a limited portion of the solar disk.

We tested various limits from 100% of the solar radius ($R_\odot$) down to 60% of the $R_\odot$ by excluding those detections that have their center outside of these limits. We plot the three main metrics for the tracking runs with the various limit values, where MOTA is in Figure 6.4(a), Mostly Tracked is in 6.4(b), and Identity Switches are in 6.4(c). We included the results for the 80% $R_\odot$ limit tracking in Table 6.2. The 80% limit was chosen because, as seen in Figure 6.3, most of the coronal hole mislabeling happens outside of this area. It was also chosen because it is the midpoint of our evaluations from 60% to 100% of $R_\odot$.

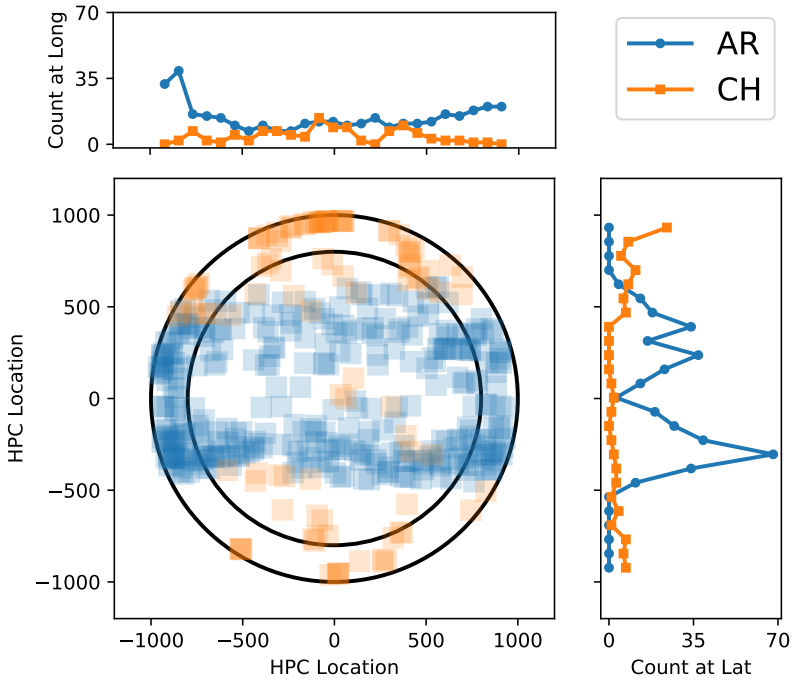
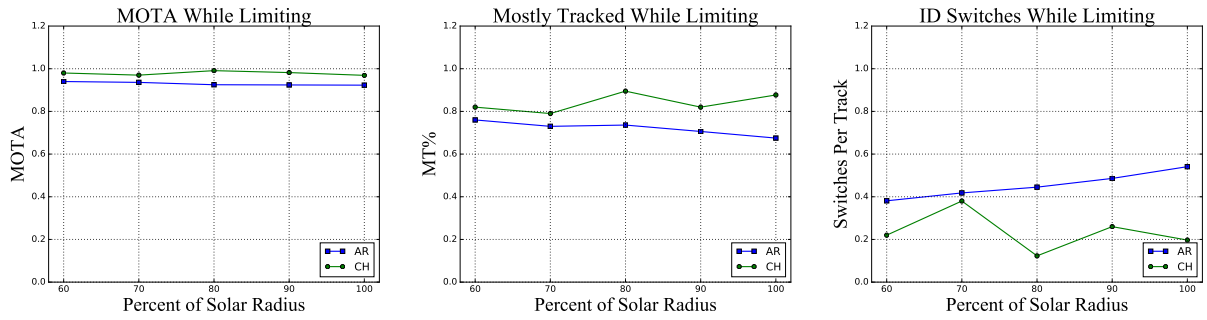


Figure 6.3: Plot showing the location where a detection is labeled as linking to a different detection by the tracking algorithm than what human labels did. The plot includes both active regions (AR) and coronal holes (CH). There are also histograms along both the Latitude and Longitude axes which show how many detections were different at a particular location. The two concentric circles are 100% $R_\odot$ (outer) and 80% $R_\odot$ (inner). Each box in the plot marks the center of the detection that has the mislabeled next detection.

In Table 6.2, we can see that the spatial limiting does improve the results of our tracking algorithm on both coronal holes and active regions. If we look at the results for coronal holes specifically, we see that the improvements are more significant for our tracking algorithm than they were for the SPoCA module. The improvements for the results of our tracking were so improved, relative to SPoCA, that we now see that our method outperforms the SPoCA results on our human-labeled dataset.

6.2 Whole Image Query Results and Analysis

In this section, we will discuss several different ways that we investigate the results obtained when using our max pooled vector image descriptors, as well as a comparison with the original image parameter space. We begin by providing a more detailed discus-



(a) (b) (c)

Figure 6.4: (a) Displays MOTA for tracking when it is limited to a percent of $R_\odot$. (b) Displays the percentage of Mostly Tracked (MT%) ground truth tracks when tracking is limited to a percent of $R_\odot$. (c) Displays the number of Identity Switches per track when tracking is limited to percent a of $R_\odot$.

sion of the results of the 1,000 random queries used for the distance measure selection process in Section 6.2.1. We then discuss the visual similarity of images in Section 6.2.2, the distribution of the most and least similar images in our data in Section 6.2.3, and provide an example of a query with its result in 6.2.4.

6.2.1 *Nearest Neighbor Distributions*

In Section 5.3, we presented the query results of the first nearest neighbor as a way for us to find the most relevant similarity (distance) measure. We then provided the sum of the temporal distance from the query in Figures 5.1(a) and 5.1(b) as our results and concluded that the $p = \frac{1}{8}$ fractional distance measure was the apparent best choice for our pooled vector descriptors. Similarly, we observed that the $p = 1$ distance measure was the apparent best choice when using the original image parameters alone.

In Figures 6.5(a) and 6.5(b) we look at each of the distance measures and the two different cases of either our pooled vector descriptors in Figure 6.5(a) or the original image parameters in Figure 6.5(b). In these two figures, we show distribution of distances for the 1,000 random queries described in Section 5.3. The distance displayed is that of the number of images, or six-minute steps; the result image is from the query.

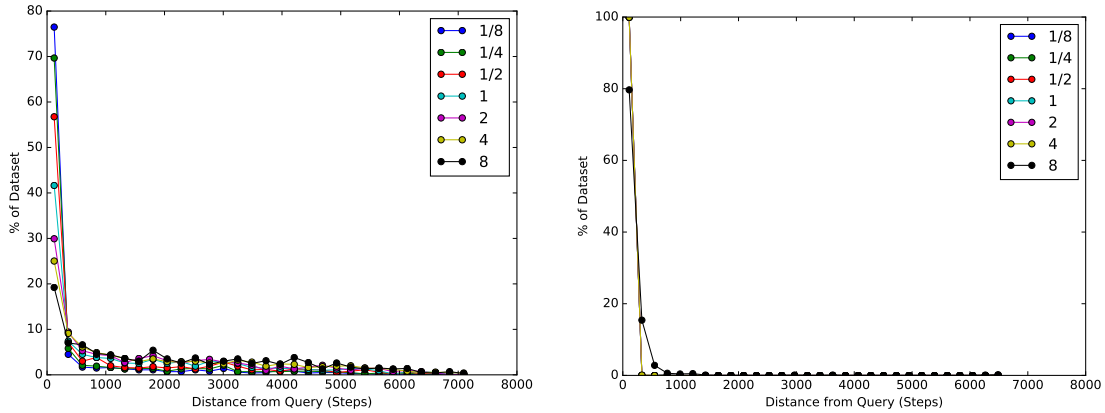
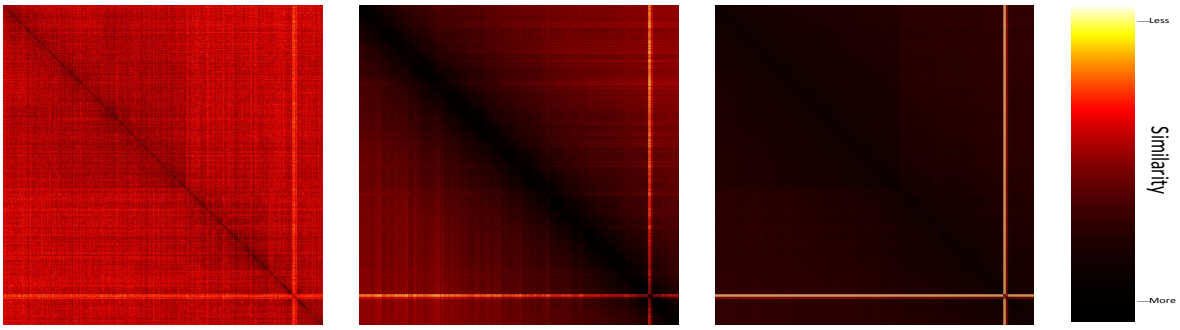


Figure 6.5: Plots showing the distribution of how temporally distant the first nearest neighbor is for every image in our dataset. 6.5(a) shows the results for pooled vectors, and 6.5(b) shows the results for image parameters.

In Figure 6.5(a), it can be seen that the highest percentage of results is very temporally near the query image for every distance measure. This result is expected given the fact that the data is essentially a video of an object that, on average, has smooth transitions from one frame to the next. However, the Sun does have periods when visual characteristics rapidly change; thus, the most similar images may not be the immediate temporal neighbor. Given this fact, it is not surprising to see that somewhere in the range of 20-25% of our results are not the immediate neighbor for our assumed best measure ($p = \frac{1}{8}$).

In Figure 6.5(b), however, we see that nearly every query result is the immediate temporal neighbor. Though this might seem to be a desirable result at first, as mentioned in [39], the problem with using a very high-dimensional vector of image parameters simply concatenated together, is that every image begins to look similar to every other image. Also, as mentioned in the previous paragraph, the sun has periods of rapid change in visual characteristics, and this method may be missing many of these changes, especially if they are not a dramatic departure from the previous image.

These observations lead us to reason that our proposed pooled vector descriptor is behaving in a manner that would be expected, given the data we are working with. It is



(a) (b) (c)

Figure 6.6: Similarity plots for 131Å where p was determined from Figure 5.1(a) and 5.1(b). Here 6.6(a) is constructed from pooled vectors using $p = \frac{1}{8}$, 6.6(b) is constructed with a single image parameter (mean) using $p = 1$, and 6.6(c) uses all image parameters and $p = 1$ on the distance measure.

returning the immediate temporal neighbor as the most similar on a large percentage of queries, but it does not fall victim to the problem of seeing only the immediate temporal neighbors as the most similar, as is seen when using the naïve descriptor of the combined image parameters.

6.2.2 *Visual Similarity*

Next, we look at a subset of the images in our dataset, and plot how similar each image is to every other image in the subset. We take the images in the 131Å wavelength over the date range of January 20, 2012 to January 23, 2012, as was done in [39]. The plots are symmetric, meaning the line down the diagonal is the comparison of each image to itself, and moving off the diagonal is comparing the image at the diagonal to either its earlier temporal neighbor (left/up) or its later temporal neighbor (right/down). The plots are seen in Figure 6.6, and the colors range from black being the most similar, according to the distance measure used, to white being the most dissimilar.

As can be seen in Figure 6.6(a), our pooled vector shows the immediate temporal neighbor as being the most similar for a large percentage of images, as would be expected from our previous results and the fact that the dataset is of a video sequence

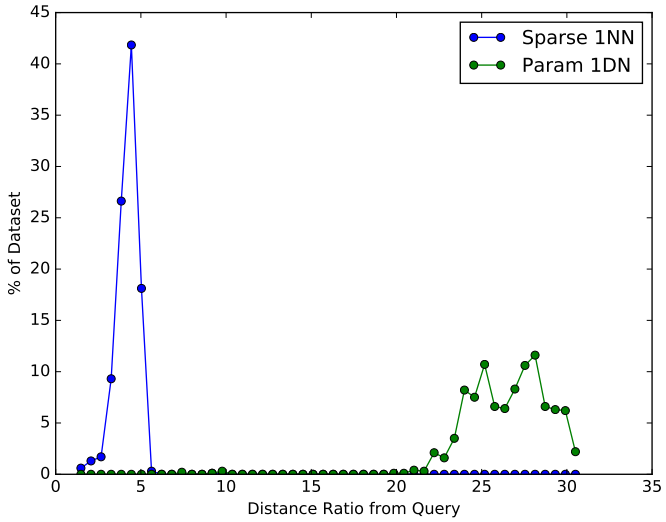


Figure 6.7: Distribution of nearest neighbor queried using the pooled vector method (Sparse 1NN) as a ratio of the distance of the nearest neighbor using the parameters. Also, the most distant neighbor using parameters (Param 1DN) as a ratio of the distance of the nearest neighbor using parameters to show the contrast.

of a smoothly changing process. In Figure 6.6(b), we display the results from just one image parameter, and as seen in [39], there is a large number of temporal neighbors that look very similar, which is only compounded in 6.6(c) when we add the other 9 image parameters.

It should also be noted that, although our pooled vector method departs rather quickly from seeing the temporal neighbors as similar, it does still have a large enough dynamic range in its comparisons to pick up the rapid changes in image composition. For example, as was noted in [39], the lower right corner of each of the plots shows a small set of images that are very dissimilar to every other image in the subset. This is due to the occurrence of a large (X-class) solar flare at that time, which any comparison method should be able to recognize as having a substantially different look than its immediate temporal neighbor. This is yet another piece of evidence that leads us to reason that our method is working correctly.

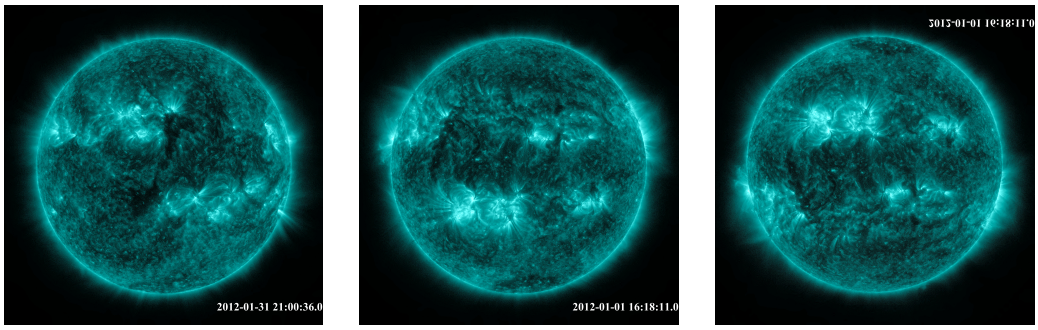
6.2.3 *Distribution of Distances*

In Figure 6.7, we examine how the distance of the most and least similar images are distributed. We construct this figure by using the 1,000 random queries described earlier. To do so, we begin by finding the first nearest neighboring image when using our pooled vector descriptor. Then, we find both the nearest and most distant images from the query image using the original image parameter space. After obtaining the neighbor results, we compute the distance of all three of the images (from the query image) in the original image parameter space.

Using the distances in the original parameter space for the three query results described above, we compute the ratio of the distance for the nearest neighbor found using the pooled vector descriptor to that of the nearest neighbor found using the original parameter space, and label it as "Sparse 1NN" in Figure 6.7. Similarly, we compute the ratio of the distance for the most distant neighbor found using the original parameter space to the nearest neighbor found using the original parameter space, and label it as "Param 1DN" in Figure 6.7. Note that both ratios use the same denominator, the distance of nearest neighbor found using the original parameter space, as we want to show how our descriptors compare to the parameter space results. If we saw an overlap between what our method says is a similar image and what the ground truth says is a distant image, then it could be argued that that our results are incorrect. However, the fact that there are two distinct distributions in this plot is yet another indication that our pooled vector descriptor indeed produces valid results.

6.2.4 *Query Example*

Finally, in Figure 6.8, we present a query example from our dataset. In this figure, we searched the month of January 2012 for the query image with the most temporally distant nearest neighbor, when using our pooled vector descriptor. The query image



(a) (b) (c)

Figure 6.8: Results of searching January 2012 for the image with the most temporally distant first nearest neighbor. 6.8(a) query image, 6.8(b) result image, 6.8(c) result flipped on vertical axis to show a more similar perspective.

is 6.8(a) from January 31, 2012 at 21:00:36, and its result is 6.8(b) on January 1, 2012 at 16:18:11. We flipped the image along the vertical axis in 6.8(c) to show a more similar perspective, as the pooling method produces a descriptor vector that is invariant to where in the image similarities occur and is simply looking for the images that have the most similar overlapping windows that cover $\frac{1}{8}$ of the image at a time.

6.3 Image Region Query Results and Analysis

In this section, we present the results of several different methods of evaluating our sparse vector region descriptors. In these evaluations we start with the same region of interest dataset that was produced for [29], and provide a comparison between the methods used in that work and our sparse vector methods. Below, we first provide a description of the evaluation dataset, which also includes a brief description of the previous representation methodology for the regions within the dataset. Finally, we provide our comparison of results for the two competing methods.

6.3.1 *Region Labeling*

As was previously mention in Section 2.3, a multi-institutional team called the Feature Finding Team (FFT) created a number of software modules to process data coming from the SDO. These modules create reports of solar phenomena (events) that are of great interest to the solar physics research community. As such, the metadata they create is directly reported to the Heliophysics Event Knowledgebase (HEK) [36]. In Figure 2.5, we provided a visualization of what these events look like when overlaid on an image from the SDO.

Similar to tracking, we utilize these reported events as identifiers for spatiotemporal regions of interest for study. Our dataset for evaluation started in [29], where two event types, Active Region (AR) and Coronal Hole (CH), were retrieved from the HEK for the year of 2012 and were utilized as the input dataset. This equated to 13,518 AR events and 10,780 CH events, with both event types having a reporting cadence of approximately four hours. In an effort to produce a balanced dataset, we compiled a list of all unique AR reports, and then each one was processed to find at least one CH event within ± 60 minutes. If no CH event was found to be in that search window for the selected AR event, then the AR report was discarded so that it can be guaranteed that there would be both event types for each timestep.

After the aforementioned filtering steps were completed, we then created a third artificial event type that was labeled as Quiet Sun (QS), which represented areas of the solar disk where neither AR nor CH events exist. To create these artificial events, the bounding box of each AR event in the filtered set was placed randomly within the solar disk, taking care to not overlap other AR or CH events at that report time. By using AR events as templates, we were attempting to replicate the natural distribution of event sizes and frequencies when creating the same number of QS events (13,518 QS objects). It is the combination of these three event types, Active Region (AR), Coronal Hole (CH),

and Quiet Sun (QS), that was used as a dataset for the work in [29]. This dataset is also the initial dataset we use for our comparisons here.

In addition to the dataset produced in [29], we utilize a second, more difficult, dataset in our evaluations of our sparse vector region descriptors. In this second dataset, we begin with the data produced by [29] and add two more solar event types to the dataset. These data types are Sun Spots (SS) and Sigmoids (SG). Both of these data types are pulled from the same HEK repository that the initial dataset was obtained from. These two data types are also from the 2012 calendar year, so as to overlap with the dataset produced previously. This produces 7,805 SG events and 3,417 SS events over the data period.

Statistical Region Description

In order to produce a descriptor for the ROIs produced by the aforementioned solar events, [29] first starts with the bounding box of the solar events. Then, utilizing the image parameter dataset that was described in Section 2.1, the image parameter cells from within the bounding box of a specific solar event are processed to create a 7-statistic summary for each parameter over all the cells within the bounding box. The statistics are ordered as: minimum, 1^{st} quartile, median, 3^{rd} quartile, maximum, average, and standard deviation, which we notate as the vector: $\{q0, q1, q2, q3, q4, avg, std\}$. This then gives each solar event a 630-dimensional vector $(9 \times 10 \times 7)$ as its representation.

Recognizing the need to perform some dimensionality reduction, [29] then utilized the Top-K feature ranking method, previously described in Section 3.1, to rank and select the top features from the 630-dimensional vector space. In this use, the different classes of solar events were used for the F-Statistic calculation, and each one of the statistic values were evaluated as individual features. In the experiments below, when using the statistical descriptor vectors for comparison, we utilize the Top-K features from this ranking method. Our ranking was performed using the average F-Statistic from 10

repetitions of a randomized undersampling of the solar events from each class over the first three months of 2012. The undersampling was applied because the number of solar events are not perfectly balanced.

6.3.2 Classification

In these evaluations, all classifications are performed using a Naïve Bayes classifier; as was found in [29], this classifier works as well as more sophisticated models such as SVM for our classification task. We begin our evaluation of our sparse vector region descriptors by utilizing them in a classification task and comparing them to the results produced by the previous methods of [29]. In the classification task, we first split the event types into the months in which they occur and take $\frac{2}{3}$ of the months (8 months) and place them in the training set. The remaining $\frac{1}{3}$ of the months (4 months) are placed in the testing set. Since the sets of events within each of these training and testing sets are not equal, we perform a random undersampling of the larger numbered events to get a balanced set of event types. This undersampling process is also performed on the testing set to have a balanced number of test cases for each of the classes.

Once the balanced sets of training and testing events are produced, the model is then trained using the training set, and then testing is performed using the testing set. This process of picking $\frac{2}{3}$ of the months for training and $\frac{1}{3}$ of the months for testing, followed by the random undersampling in each set, is repeated for each combination of 8 months training and 4 months testing available in our dataset or $\binom{12}{4} = 495$ combinations of different testing months. The results that we present are the mean values seen from these $\binom{12}{4}$ combinations of the training and testing processes. This process is the same regardless of the region descriptor vector type we are processing.

In the following subsections, we present the results and comparisons between our sparse vector region descriptor and the results produced by the previous statistic vector methods of [29]. We begin with the first dataset of 3 classes; Active Region (AR), Coronal

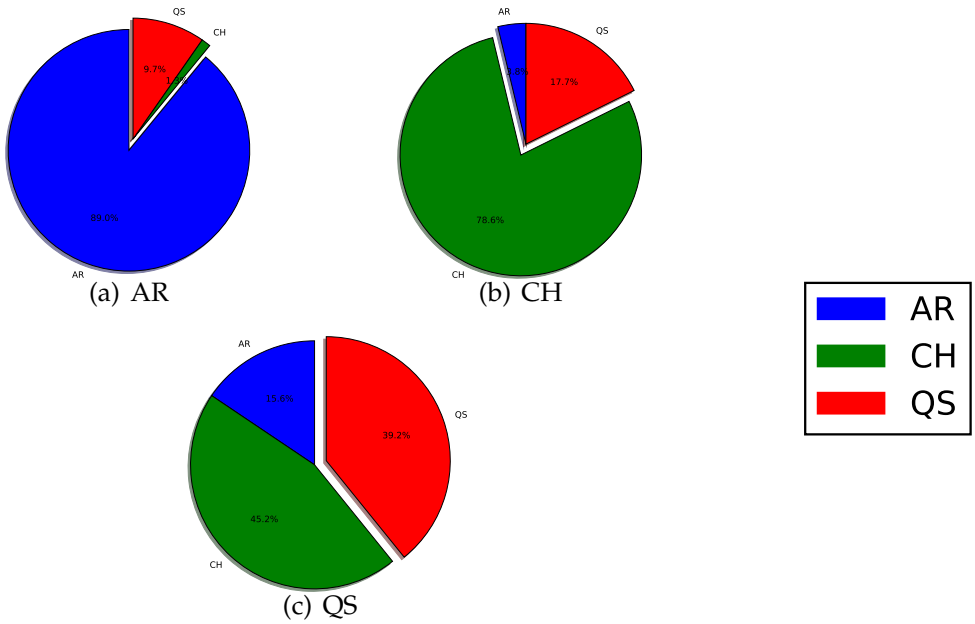


Figure 6.9: Plots showing the classification results on three classes using the statistical description methods of [29]. 6.9(a) shows how the events in AR class were classified. 6.9(b) shows how the events in the CH class were classified. 6.9(c) shows how the events in the QS class were classified.

Hole (CH), and Quite Sun (QS). Then we present the results of the more difficult 5-class dataset that contains the additional event types of Sigmoid (SG) and Sun Spot (SS).

3-Class Dataset

As stated above, we begin with the original dataset produced by [29] that contains the three event types Active Region (AR), Coronal Hole (CH), and Quiet Sun (QS). In order to produce a fair comparison between the statistic feature method from [29] and our sparse vector method, we evaluated several sizes of input statistic feature vector to our classification task and chose the best performing results to discuss here. In our evaluation of these different sized statistic feature vectors, we found that a vector of the top-12 rated statistic features worked the best.

The pie charts in Figure 6.9 show the average classification breakdown of the statistical feature method from [29]. From the figures, we can see that the statistic features do a relatively decent job at describing the AR and CH classes in such a way that the classifier

can differentiate between the two. In 6.9(a), we see that the AR class has a relatively decent average accuracy of about 89%, with a majority of the misclassifications going to the QS class. Then, in 6.9(b), we see that the CH class has an average accuracy of about 78.6%, again with a majority of the misclassifications going to the QS class. In both of these two subfigures, we see very few AR being classified as CH (1.3%) or CH being classified as AR (3.8%), which should be the case as these two event types are so visually dissimilar, as ARs are seen as bright spots in some wavelengths and CHs are relatively dark in some wavelengths.

However, in 6.9(c), we see that the statistical method has some difficulties correctly classifying the QS objects, the the extent that they are incorrectly classified as CH objects more often than they are correctly classified as QS objects, at 45.2% CH vs. QS 39.2%. This drags on the overall classification accuracy and brings the statistical feature method's overall classification accuracy rate to 71.1%. This difficulty probably arises due to the fact that the CH and QS event types are not as drastically dissimilar as the AR and CH event types are. This leads to the the conclusion that the statistical method will have even more difficulty when more classes that are more closely related are added in the 5-class dataset.

AS was done for the statistical features, we evaluated several settings for our sparse vector region descriptors and utilized the best results, which are shown in Figure 6.10. The settings that were chosen are that of a dictionary with 40 elements in it and an input vector composed of all 90 image parameters. With these settings, the sparse vector method sees an 80.2% overall classificaton accuracy rate, a 9.1% improvement over the statistical feature method.

Similar to the results seen when using the statistical features, Figure 6.10 shows that the sparse features describe the AR-class objects in a manner such that the classifier can produce even better results than before. As seen in 6.10(a), the AR class has a 93.1% correct classification average, and a majority of the misclassifications were seen as QS

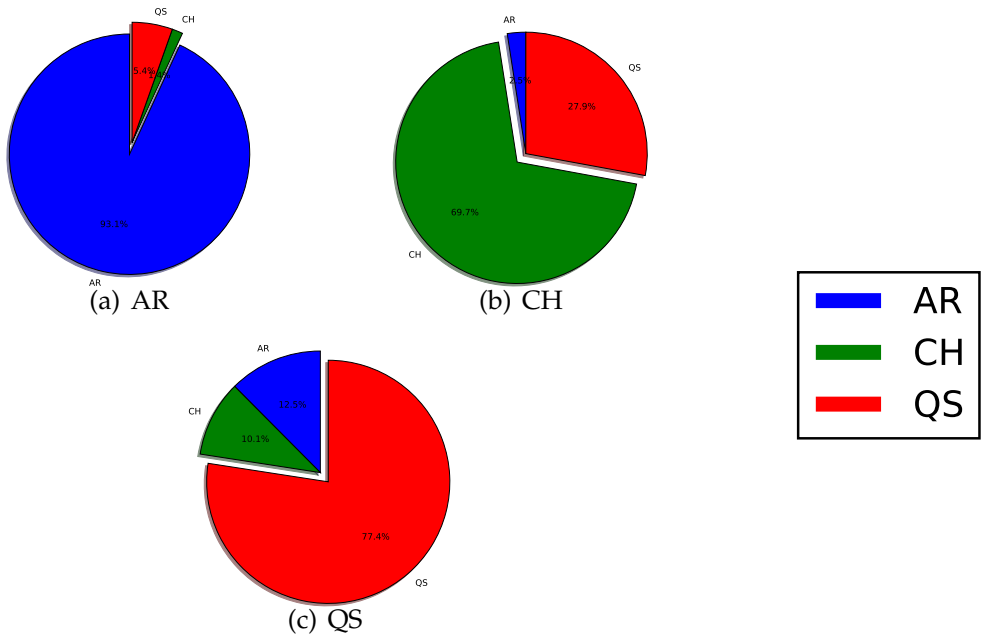
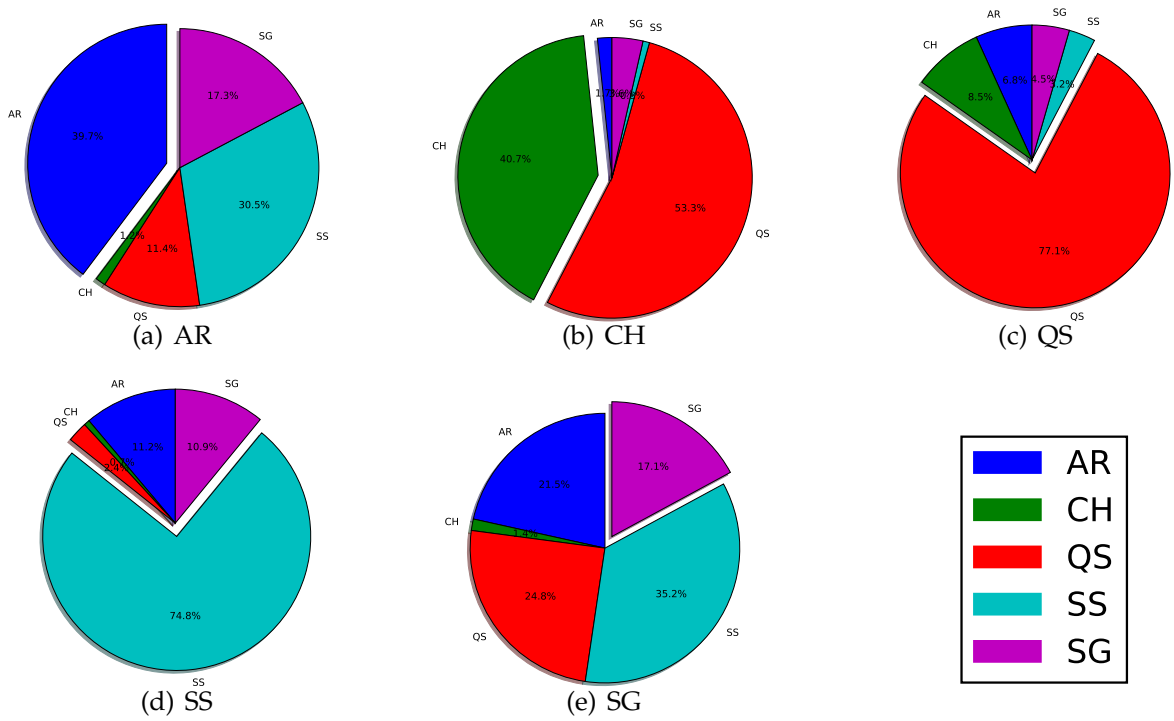


Figure 6.10: Plots showing the classification results on three classes using the our sparse coding region descriptor method. 6.10(a) shows how the events in AR class were classified. 6.10(b) shows how the events in the CH class were classified. 6.10(c) shows how the events in the QS class were classified.

objects. In 6.10(b), we see that the sparse vector method leads to a slightly lower correct classification average for CH objects at 69.7%, which is a decline of 8.9%. Though it did decline, the increase in incorrect labels mostly went to QS event type, which is more acceptable than had it gone to the AR type as they are less similar.

Where we see the greatest divergence from the statistical feature method is on the QS events in 6.10(c). Here we see that the sparse vector method produces a 77.4% average classification accuracy, which is a 38.2% improvement over the statistical method for this class of events. This improvement leads us to postulate that the sparse vector method will be more adept at differentiating between the more similar event types when we introduce the SG and SS event types in the next section. This shall be important since the SG and SS event types tend to co-occur with the AR event types making their visual similarity quite pronounced in some wavelengths.



(a) AR (b) CH (c) QS

(d) SS (e) SG

Figure 6.11: Plots showing classification results on five classes using the statistical description methods of [29]. 6.11(a) shows how the events in AR class were classified. 6.11(b) shows how the events in the CH class were classified. 6.11(c) shows how the events in the QS class were classified. 6.11(d) shows how the events in the SS class were classified. 6.11(e) shows how the events in the SG class were classified.

5-Class Dataset

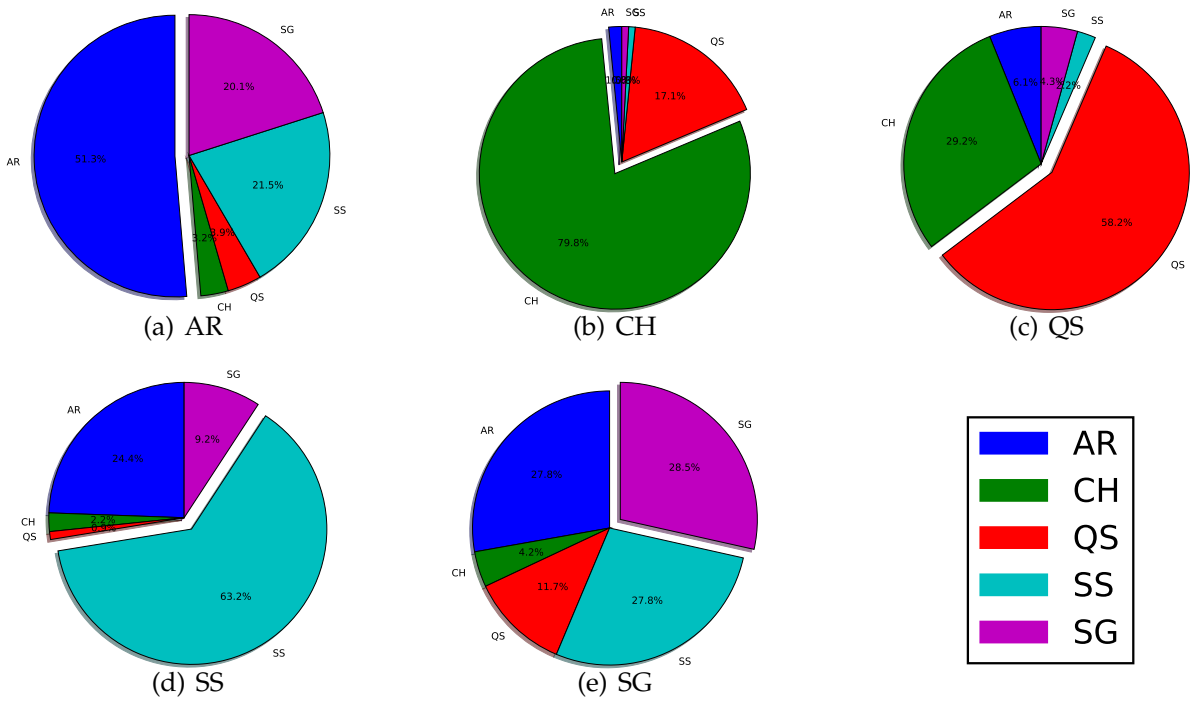
After seeing the results from the 3-class dataset, we concluded that it would be best if we investigated a more difficult dataset. So, as stated in Section 6.3.1, we added two additional data types to our original dataset produced in [29]. This 5-class dataset contains the three event types Active Region (AR), Coronal Hole (CH), and Quiet Sun (QS) from before and the additional two event types of Sun Spot (SS), and Sigmoid (SG). We then reproduced the experiments from the 3-class dataset and present them here. Just as was done with the 3-class dataset, we evaluated several sizes of input statistical feature vectors as input to classification task and chose the best performing results to discuss here. In our evaluation of these different sized statistical feature vectors, we found that a vector of the top-199 rated statistical features worked the best.

Like with the 3-class dataset, we produced Figure 6.11 showing the average classification breakdown of the statistical feature method. From the figures, we can see that the statistical features still do quite well at differentiating between AR and CH events, with few of either one being classified as the other. They also do quite well with providing a reasonable descriptor for the QS objects, giving an average classification accuracy of 77.1%. This was unexpected judging from the results found in the 3-class dataset, and could be because of the more numerous features included in the models used here. In addition, we see that the statistical features do remarkably well with one of the event types added for this dataset, with SS achieving an average of 74.8% classification accuracy.

However, the statistical features are beginning to have difficulties describing the AR in such a way that it can be differentiated from the two new event types SS and SG, which is what we were expecting, as the SS and SG event types co-occur with the AR event type. Yet another issue that can be seen in the figure is that the CH classes are now being classified as a QS object more often than the correct event type, though not as often as any of the new event types. One final issue with the statistical feature method is that the SG event type is classified as one of three incorrect classes more than it is identified correctly at only 17.1% of the time. We can see that the classes AR, SS, and QS, are picked more often than the correct SG label. In all, these mixed results lead to an overall average of 49.9% classification accuracy for the 5-class dataset.

Next, like was done for the statistical features, we evaluated several settings for our sparse vector region descriptors and utilized the best results in Figure 6.12. The settings that were chosen are that of a dictionary with 32 elements in it, and an input vector composed of all 90 image parameters. With these settings, the sparse vector method sees a 56.4% overall classificaton accuracy rate, a 6.5% improvement over the statistical feature method.

In the figure, we can see that the AR, CH, and SG classes see improvements over the statistical feature method, with CH seeing the most marked improvement. In 6.12(b),



(a) AR (b) CH (c) QS

(d) SS (e) SG

Figure 6.12: Plots showing the classification results on five classes using the our sparse coding region descriptor method. 6.12(a) shows how the events in AR class were classified. 6.12(b) shows how the events in the CH class were classified. 6.12(c) shows how the events in the QS class were classified. 6.12(d) shows how the events in the SS class were classified. 6.12(e) shows how the events in the SG class were classified.

the CH events go from being more likely to be classified as a QS event than a CH event, to being more likely to be classified a CH event than all others combined. We also see AR improving to be more likely to be classified as an AR event than all others combined in 6.12(a). Though we do see some drop in average classification accuracy on QS and SS event types, the fact that all five event types are now more likely to be classified as the correct event type than any one other event type bodes well for the possibility of using k-nearest neighbors (KNN) as a classification method, even though our focus on KNN in the next section is not for the classification task.

6.3.3 Retrieval

Lastly, as was done in [29], we look at k-nearest neighbor (KNN) retrieval results on the sparse vector descriptors to assess CBIR capabilities and specifically Region-based

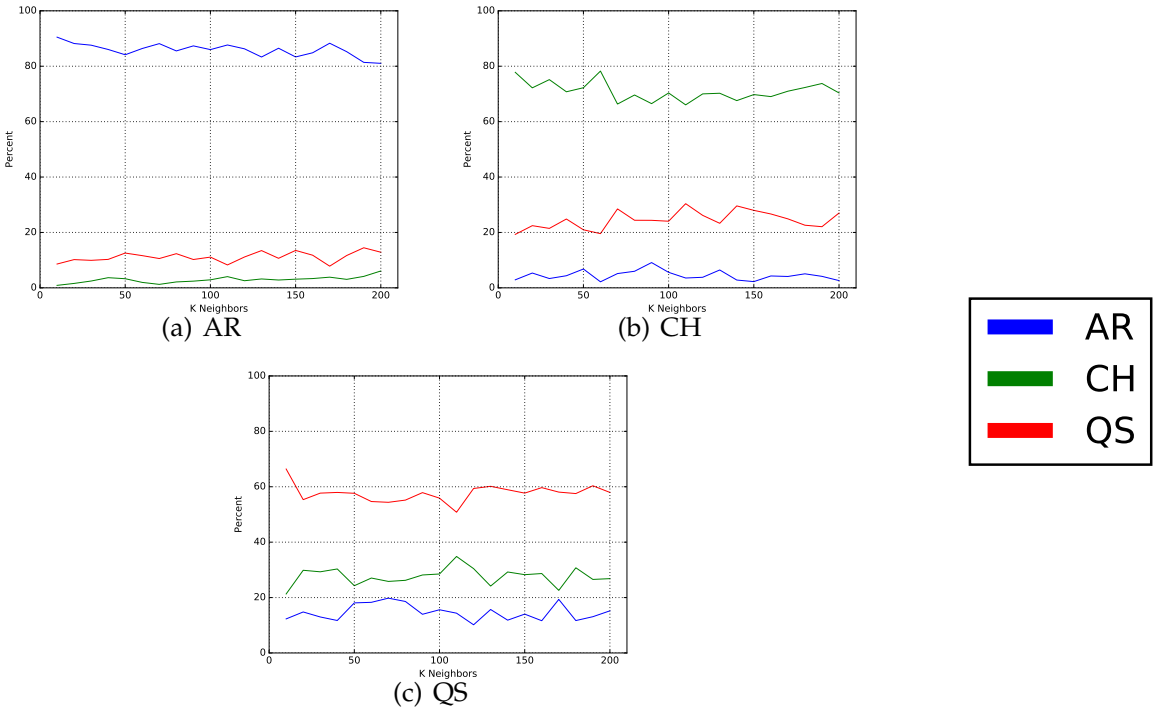
Querying (RBQ) performance. As was done in [29], in order to obtain this set of results, we select 100 random data instances and retrieve the nearest N neighbors using the Euclidean distance on our sparse vector descriptors. The dimension of the vector depends on the dataset we utilize. We use the aggregate of the 100 random instance queries for the results to calculate the mean and display the results for a single instance of N in the KNN query. Just as was done for the classification task, we utilize both our 3-class dataset from [29], and the augmented dataset with 5 classes.

Unlike the classification evaluations performed above, the RBQ retrieval process uses the entire unbalanced dataset, which provides an evaluation that is closer to what would be seen in a real-world application. As was noted in [29], in both our 3-class and 5-class datasets, there are many more events per class than our largest query, so we do not need to be concerned with a query exhausting the dataset. For example, the smallest collection of any one event type is the SS event type, which has 3,417 instances for a query of $K = 200$ to find. Below we discuss the results seen for each of these two datasets.

3-Class Dataset

We begin our KNN evaluation with the 3-class dataset that covers the calendar year of 2012, and contains the AR, CH, and QS classes. In Figure 6.13 we preset the average percentage of neighbor class membership obtained from the 100 randomly selected query instances. The graphs in this figure show the results as we vary the number of neighbors returned for each set of 100 queries from $K = 10$ to $K = 200$ in increments of 10 neighbors.

From Figure 6.13(a), we can ascertain that the AR class is clearly distinguishable from the CH and QS classes, as over 80% of the returned neighbors are of the AR class, even for queries of $K = 200$. In Figure 6.13(b), we find similar results with the CH class, though to a lesser extent than with the AR class. In it, we see that the CH class has between 70% and 80% of the returned neighbors of the same class as the query event. In



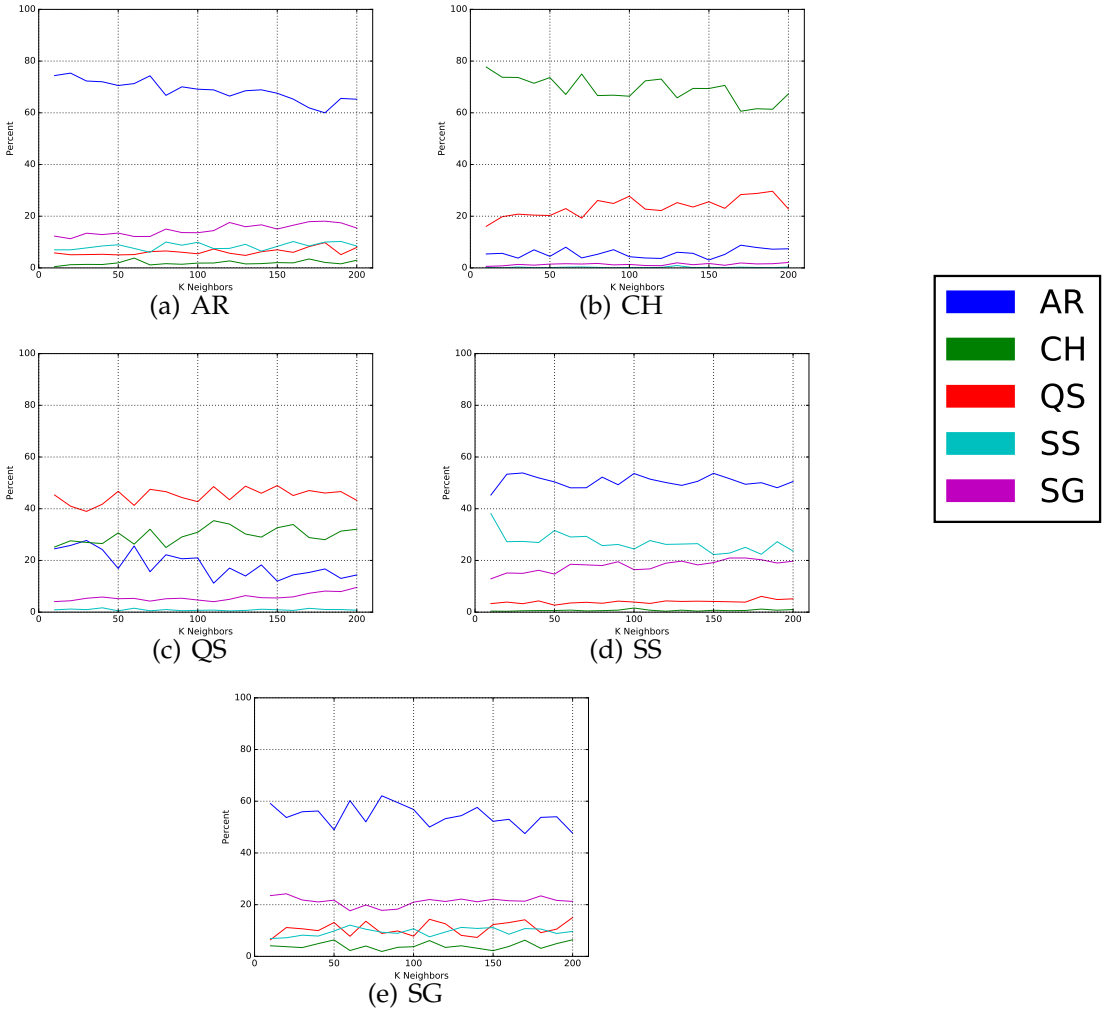
(a) AR (b) CH (c) QS

Figure 6.13: Plots showing the percentage of neighbors of each class type for KNN queries on the 3-class dataset using the our sparse coding region descriptor method. 6.13(a) shows the neighbors for the AR class. 6.13(b) shows the neighbors for the CH class. 6.13(c) shows the neighbors for the QS class.

Figure 6.13(c), we see the QS class again being the more difficult class to find definable characteristics in, as the percentage of neighbors being returned from the queries hovers around the 60% line. Though it would be desirable for QS event type results to be nearer the percentage averages seen for the AR or CH event types, the observed results are not entirely unexpected given the random process used to create the QS events.

5-Class Dataset

Next we move onto evaluating the KNN retrieval using the sparse vector descriptors on our augmented 5-class dataset, which includes the two additional event types of SG and SS. As was done for the 3-class dataset, we present the average percentage of neighbor class membership obtained from the 100 randomly selected query instances in Figure 6.14. Again, the graphs in this figure show the results as we vary the number of



(a) AR (b) CH (c) QS (d) SS (e) SG

Figure 6.14: Plots showing the percentage of neighbors of each class type for KNN queries on 5-class dataset using the our sparse coding region descriptor method. 6.14(a) shows the neighbors for the AR class. 6.14(b) shows the neighbors for the CH class. 6.14(c) shows the neighbors for the QS class. 6.14(d) shows the neighbors for the SS class. 6.14(e) shows the neighbors for the SG class.

neighbors returned for each set of 100 queries from $K = 10$ to $K = 200$ in increments of 10 neighbors.

In Figure 6.14(a), we can see that, for the AR class, introducing the two new classes to the dataset has reduced the percentage of returned neighbors that are the correct class by around 10% from the results seen with the 3-class dataset. It is not entirely surprising that we see some additional responses coming from the SG and SS classes, as they co-occur with the AR class and therefore will have highly similar visual characteristics. Also not surprisingly, we do not see much of a shift in the results of the CH class in Figure 6.14(b), as the additional SG and SS classes should not have overlapping visual characteristics with CH events and therefore should not be very similar to them. In Figure 6.14(c), we see a similar reduction to that which was seen in the AR class, with a majority of the increased incorrect events being from the SG class. This is probably due to the significant spatial overlap between the SG events and AR events.

At first glance, the results in both Figure 6.14(e) and Figure 6.14(d) look somewhat problematic, with both returning the AR class as the closest neighbor more often than events of the same event type. However, given that they both co-occur with the AR class, and that there are so many more AR events than either the SG or SS event type, it is not such a surprising result to see. When we couple this with the fact that the event type second most likely to be returned in these cases is the same as the query event type, we can conclude that the sparse vector descriptor is working relatively well.

Additionally, for the two problem event types of SG and SS, we can point out that event types that the query event do not have much visual similarity with, like the CH event type, do not show up in the returned neighbor set with great frequency. For example, in 6.14(d), it is difficult to even discern the CH response frequency as it rarely leaves the zero axis. Similarly, in 6.14(e), the CH event type is the least likely of all the event types to be returned. The combinations of all these factors, gives us confidence in the results returned by this method.

7 CONCLUSION

This dissertation has presented our examination of utilizing sparse coding to construct a variety of methods for comparing both whole SDO AIA images and regions thereof. We have presented numerous results showing the effectiveness of these methods for tracking various solar events, as well as for producing efficiently comparable vectors for use in content-based image retrieval. In using these methods for tracking, we were able to improve upon our previous tracking work so that the model of appearances was trained online and can be applied to solar event types that do not have previously compiled tracking information. Then, in whole SDO AIA image and regions of image comparisons, we were able to improve the selectivity of KNN retrieval results, while also utilizing a distance function that is a metric and therefore applicable for indexing.

7.1 Future Work

We plan to extend our research on using sparse vector descriptors for whole and regions of images querying by working towards an online system that continuously integrates new images as they become available. A key component to being able to achieve the integration of a continuously updated dataset will be the use of an indexing schema that can be updated without having to recalculate the distance between every image in the dataset. To accomplish this, we plan to utilize the iDistance index investigated in [73] and [74] as the indexing structure that will accommodate our needs.

Furthermore, given the fact that there is a large, and growing, amount of video data available from the SDO mission, and that solar phenomena are not static single-frame occurrences, these approaches of single similar image retrieval are of limited use to solar physics researchers. In order to be more useful tool, a retrieval system should allow

a researcher to retrieve a sequence of images that are similar to a query sequence of images that depict phenomena of interest. Once a system of single image retrieval is produced, it is our intent to make the first steps towards producing a retrieval system that would be capable of this sequence retrieval task. To do this, we must first develop a summarization technique to quickly and concisely represent sequences of images.

Summaries exist in many different types; there are summaries of collections of documents [75], summaries of video [76,77], and navigation summaries of robotic motion [78], just to name a few. These are useful things in that they can briefly and concisely present the data that is being summarized in a significantly smaller footprint than the original. We intend to start our investigation of producing video summaries by following the work of [77], in that we will extract a global feature from each image, perform a hierarchical clustering on these feature vectors, and finally, reduce redundancy through filtering by the resultant cluster centers. Our approach will differ somewhat from [77] in that our global feature set for an image shall not based on the HSV histogram of an image, but shall use sparse coding and a pooling pyramid, such as multi-scale max pooling, as is described in [65] and [66] to represent each image.

BIBLIOGRAPHY

[1] S. S. Blackman. Multiple hypothesis tracking for multiple target tracking. *IEEE Aerospace and Electronic Systems Magazine*, 19(1):5–18, Jan 2004.

[2] Stefan Berchtold, Christian Böhm, and Hans-Peter Kriegal. The pyramid-technique: Towards breaking the curse of dimensionality. *SIGMOD Rec.*, 27:142–153, 1998.

[3] Michael Schuh, Tim Wylie, and Rafal Angryk. Mitigating the curse of dimensionality for exact knn retrieval, 2014.

[4] George L. Withbroe. *Living With a Star*, pages 45–51. American Geophysical Union, 2013.

[5] National Research Council. *Severe Space Weather Events–Understanding Societal and Economic Impacts: A Workshop Report*. The National Academies Press, Washington, DC, 2008.

[6] D. H. Boteler. Geomagnetic hazards to conducting networks. *Natural Hazards*, 28(2):537–561, 2003.

[7] W. Dean Pesnell, B. J. Thompson, and P. C. Chamberlin. The solar dynamics observatory (sdo). *Solar Physics*, 275(1):3–15, 2011.

[8] P.C.H. Martens, G.D.R. Attrill, A.R. Davey, A. Engell, S. Farid, P.C. Grigis, J. Kasper, K. Korreck, S.H. Saar, A. Savcheva, Y. Su, P. Testa, M. Wills-Davey, P.N. Bernasconi, N.-E. Raouafi, V.A. Delouille, J.F. Hochedez, J.W. Cirtain, C.E. DeForest, R.A. Angryk, I. De Moortel, T. Wiegelmann, M.K. Georgoulis, R.T.J. McAteer, and R.P. Timmons. Computer vision for the solar dynamics observatory (sdo). *Solar Physics*, 275(1-2):79–113, 2012.

[9] T. N. Woods, F. G. Eparvier, R. Hock, A. R. Jones, D. Woodraska, D. Judge, L. Didkovsky, J. Lean, J. Mariska, H. Warren, D. McMullin, P. Chamberlin, G. Berthiaume, S. Bailey, T. Fuller-Rowell, J. Sojka, W. K. Tobiska, and R. Viereck. Extreme ultraviolet variability experiment (eve) on the solar dynamics observatory (sdo): Overview of science objectives,

instrument design, data products, and model developments. *Solar Physics*, 275(1):115–143, 2012.

[10] P. H. Scherrer, J. Schou, R. I. Bush, A. G. Kosovichev, R. S. Bogart, J. T. Hoeksema, Y. Liu, T. L. Duvall, J. Zhao, A. M. Title, C. J. Schrijver, T. D. Tarbell, and S. Tomczyk. The helioseismic and magnetic imager (hmi) investigation for the solar dynamics observatory (sdo). *Solar Physics*, 275(1):207–227, 2012.

[11] James R. Lemen, Alan M. Title, David J. Akin, Paul F. Boerner, Catherine Chou, Jerry F. Drake, Dexter W. Duncan, Christopher G. Edwards, Frank M. Friedlaender, Gary F. Heyman, Neal E. Hurlburt, Noah L. Katz, Gary D. Kushner, Michael Levay, Russell W. Lindgren, Dnyanesh P. Mathur, Edward L. McFeaters, Sarah Mitchell, Roger A. Rehse, Carolus J. Schrijver, Larry A. Springer, Robert A. Stern, Theodore D. Tarbell, Jean-Pierre Wuelser, C. Jacob Wolfson, Carl Yanari, Jay A. Bookbinder, Peter N. Cheimets, David Caldwell, Edward E. Deluca, Richard Gates, Leon Golub, Sang Park, William A. Podgorski, Rock I. Bush, Philip H. Scherrer, Mark A. Gummin, Peter Smith, Gary Auker, Paul Jerram, Peter Pool, Regina Soufli, David L. Windt, Sarah Beardsley, Matthew Clapp, James Lang, and Nicholas Waltham. The atmospheric imaging assembly (aia) on the solar dynamics observatory (sdo). *Solar Physics*, 275(1):17–40, 2012.

[12] Juan M. Banda, Rafal A. Angryk, and Petrus C. Martens. Article: Imagefarmer: Introducing a data mining framework for the creation of large-scale content-based image retrieval systems. *International Journal of Computer Applications*, 79(13):8–13, October 2013.

[13] J.M. Banda, Chang Liu, and R.A. Angryk. Region-based querying of solar data using descriptor signatures. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 1–7, Dec 2013.

[14] J.M. Banda and R.A. Angryk. Regional content-based image retrieval for solar images: Traditional versus modern methods. *Astronomy and Computing*, 13:108 – 116, 2015.

[15] J. M. Banda, R. A. Angryk, and P. C. Martens. On the surprisingly accurate transfer of image parameters between medical and solar images. In *2011 18th IEEE International Conference on Image Processing*, pages 3669–3672, Sept 2011.

[16] D.J. Kempton, M.A. Schuh, and R.A. Angryk. Towards using sparse coding in appearance models for solar event tracking. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1252–1259, July 2016.

[17] D. Kempton, K.G. Pillai, and R. Angryk. Iterative refinement of multiple targets tracking of solar events. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 36–44, Oct 2014.

[18] D.J. Kempton and R.A. Angryk. Tracking solar events through iterative refinement. *Astronomy and Computing*, 13:124–135, 2015.

[19] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. *On the Surprising Behavior of Distance Metrics in High Dimensional Space*, pages 420–434. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[20] A. Ahmadzadeh, D. J. Kempton, M. A. Schuh, and R. A. Angryk. Improving the functionality of tamura directionality on solar images. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 2518–2526, Dec 2017.

[21] Hideyuki Tamura, Shunji Mori, and Takashi Yamawaki. Textural features corresponding to visual perception. *Systems, Man and Cybernetics, IEEE Transactions on*, 8(6):460–473, June 1978.

[22] Juan Banda and R.A. Angryk. An experimental evaluation of popular image parameters for monochromatic solar image categorization. In *FLAIRS Conference, 2010 Twenty-Third*, pages 380–385, May 2010.

[23] Juan M Banda and Rafal A Angryk. Selection of image parameters as the first step towards creating a cbir system for the solar dynamics observatory. In *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on*, pages 528–534. IEEE, 2010.

[24] P.M. McInerney, J.M. Banda, and R.A. Angryk. On using sift descriptors for image parameter evaluation. In *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pages 32–39, Dec 2013.

[25] J. M. Banda and R. A. Angryk. On the effectiveness of fuzzy clustering as a data discretization technique for large-scale classification of solar images. In *2009 IEEE International Conference on Fuzzy Systems*, pages 2019–2024, Aug 2009.

[26] M.A. Schuh, R.A. Angryk, K.G. Pillai, J.M. Banda, and P.C. Martens. A large-scale solar image dataset with labeled event regions. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 4349–4353, Sept 2013.

[27] M.A. Schuh and R.A. Angryk. Massive labeled solar image data benchmarks for automated feature recognition. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 53–60, Oct 2014.

[28] M.A. Schuh, R.A. Angryk, and P.C. Martens. Solar image parameter data from the sdo: Long-term curation and data mining. *Astronomy and Computing*, 13:86 – 98, 2015.

[29] Michael Schuh, Dustin Kempton, and Rafal Angryk. A region-based retrieval system for heliophysics imagery, 2017.

[30] Ahmet Kucuk, Berkay Aydin, Soukaina Filali Boubrahimi, Dustin Kempton, and Rafal A. Angryk. *An Integrated Solar Database (ISD) with Extended Spatiotemporal Querying Capabilities*, pages 405–410. Springer International Publishing, Cham, 2017.

[31] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online Learning for Matrix Factorization and Sparse Coding. *ArXiv e-prints*, August 2009.

[32] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, Taylor & Francis Group, Boca Raton, FL, 2015.

[33] S.S. Chen, Donoho D.L., and Saunders M.A. Atomic decomposition by basis pursuit. *Scientific Computing, SIAM Journal on*, 20(1):33–61, 1999.

[34] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.

[35] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Ann. Statist.*, 32(2):407–499, 04 2004.

[36] N. Hurlburt, M. Cheung, C. Schrijver, L. Chang, S. Freeland, S. Green, C. Heck, A. Jaffey, A. Kobashi, D. Schiff, J. Serafin, R. Seguin, G. Slater, A. Somani, and R. Timmons. Heliophysics event knowledgebase for the solar dynamics observatory (sdo) and beyond. In Phillip Chamberlin, WilliamDean Pesnell, and Barbara Thompson, editors, *The Solar Dynamics Observatory*, pages 67–78. Springer US, 2012.

[37] B. Aydin, D. Kempton, V. Akkineni, R. Angryk, and K.G. Pillai. Mining spatiotemporal co-occurrence patterns in solar datasets. *Astronomy and Computing*, 13:136 – 144, 2015.

[38] Berkay Aydin and Rafal Angryk. Discovering spatiotemporal event sequences. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, MobiGIS '16, pages 46–55, New York, NY, USA, 2016. ACM.

[39] Juan M. Banda, Michael A. Schuh, Tim Wylie, Patrick McInerney, and Rafal A. Angryk. *When Too Similar Is Bad: A Practical Example of the Solar Dynamics Observatory Content-Based Image-Retrieval System*, pages 87–95. Springer International Publishing, Cham, 2014.

[40] D. J. Kempton, M. A. Schuh, and R. A. Angryk. Describing solar images with sparse coding for similarity search. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3168–3176, Dec 2016.

[41] H. Peng, Fulmi Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(8):1226–1238, Aug 2005.

[42] C. Ding and H. Peng. Minimum redundancy feature selection from microarray gene expression data. In *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 523–528, Aug 2003.

[43] I. Batal and M. Hauskrecht. A supervised time series feature extraction technique using dct and dwt. In *2009 International Conference on Machine Learning and Applications*, pages 735–739, Dec 2009.

[44] Dustin J. Kempton, Michael A. Schuh, and Rafal A. Angryk. *Towards Feature Selection for Appearance Models in Solar Event Tracking*, pages 88–101. Springer International Publishing, Cham, 2016.

[45] Thacker Neil A. Rockett Peter I Aherne, Frank J. The bhattacharyya metric as an absolute similarity measure for frequency coded data. *Kybernetika*, 34(4):[363]–368, 1998.

[46] C. Verbeeck, V. Delouille, B. Mampaey, and R. De Visscher. The spoca-suite: Software for extraction, characterization, and tracking of active regions and coronal holes on euv images. *Astronomy & Astrophysics*, 561:A29, January 2014.

[47] B. Aydin, R. Angryk, S. Filali Boubrahimi, and S. M. Hamdi. Spatiotemporal Frequent Pattern Discovery from Solar Event Metadata. *AGU Fall Meeting Abstracts*, pages SH34A–08, December 2016.

[48] B. Aydin and R. A. Angryk. Spatiotemporal event sequence mining from evolving regions. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 4172–4177, Dec 2016.

[49] B. Aydin and R. Angryk. Spatiotemporal Frequent Pattern Mining on Solar Data: Current Algorithms and Future Directions. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 575–581, Nov 2015.

[50] J. Berclaz, F. Fleuret, and P. Fua. Multiple object tracking using flow linear programming. In *Performance Evaluation of Tracking and Surveillance (PETS-Winter), 2009 Twelfth IEEE International Workshop on*, pages 1–8, Dec 2009.

[51] Li Zhang, Yuan Li, and R. Nevatia. Global data association for multi-object tracking using network flows. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008.

[52] M. Hofmann, M. Haag, and G. Rigoll. Unified hierarchical multi-object tracking using global data association. In *Performance Evaluation of Tracking and Surveillance (PETS), 2013 IEEE International Workshop on*, pages 22–28, Jan 2013.

[53] Baiyang Liu, Junzhou Huang, Lin Yang, and C. Kulikowsk. Robust tracking using local sparse appearance model and k-selection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1313–1320, June 2011.

[54] Xue Mei, Haibin Ling, Yi Wu, E. Blasch, and Li Bai. Minimum error bounded efficient ℓ_1 tracker with occlusion detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1257–1264, June 2011.

[55] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1822–1829, June 2012.

[56] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, T. Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3360–3367, June 2010.

[57] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801, June 2009.

[58] R. Donner, B. Micusik, G. Langs, and H. Bischof. Sparse mrf appearance models for fast anatomical structure localisation. In *Proceedings of the British Machine Vision Conference*, pages 109.1–109.10. BMVA Press, 2007.

[59] Xiabi Liu, Ling Ma, Li Song, Yanfeng Zhao, Xinming Zhao, and Chunwu Zhou. Recognizing common ct imaging signs of lung diseases through a new feature selection method based on fisher criterion and genetic optimization. *Biomedical and Health Informatics, IEEE Journal of*, 19(2):635–647, March 2015.

[60] Baiyang Liu, Junzhou Huang, C. Kulikowski, and Lin Yang. Robust visual tracking using local sparse appearance model and k-selection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(12):2968–2981, Dec 2013.

[61] Chee-Yee Chong. Graph approaches for data association. In *Information Fusion (FUSION), 2012 15th International Conference on*, pages 1578–1585, July 2012.

[62] C. Caballero and M.C. Aranda. Automatic tracking of active regions and detection of solar flares in solar euv images. *Solar Physics*, 289(5):1643–1661, 2014.

[63] H. Pirsiavash, D. Ramanan, and C.C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1201–1208, June 2011.

[64] M.A. Schuh, J. Banda, R. Angryk, and P.C. Martens. Introducing the first publicly available content-based image-retrieval system for the solar dynamics observatory mission. In *AAS/SPD Meeting*, volume 44 of *Solar Physics Division Meeting*, page #100.97, July 2013.

[65] Jianchao Yang, Kai Yu, Yihong Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 1794–1801, June 2009.

[66] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178, 2006.

[67] Maximilian Riesenhuber and Tomaso Poggio. Hierarchical models of object recognition in cortex. *Nature Neuroscience*, 2(11):1019–1025, 1999.

[68] D. François, V. Wertz, and M. Verleysen. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering*, 19(7):873–886, July 2007.

[69] Damien François, Vincent Wertz, Michel Verleysen, et al. Non-euclidean metrics for similarity search in noisy datasets. In *ESANN*, pages 339–344, 2005.

[70] Schuh, Michael A., Angryk, Rafal A., and Martens, Petrus C. A large-scale dataset of solar event reports from automated feature recognition modules. *J. Space Weather Space Clim.*, 6:A22, 2016.

[71] Yuan Li, Chang Huang, and R. Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2953–2960, June 2009.

[72] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics. *J. Image Video Process.*, 2008:1:1–1:10, January 2008.

[73] Michael A. Schuh, Tim Wylie, Juan M. Banda, and Rafal A. Angryk. A comprehensive study of idistance partitioning strategies for knn queries and high-dimensional data indexing. In Georg Gottlob, Giovanni Grasso, Dan Olteanu, and Christian Schallhart, editors, *Big Data*, pages 238–252, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[74] M. A. Schuh and R. A. Angryk. On the theory and practice of high-dimensional data indexing with idistance. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 3593–3600, Dec 2016.

[75] D. M. Russell and A. Dieberger. Synthesizing evocative imagery through design patterns. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 4 pp.–, Jan 2003.

[76] D. M. Russell. A design pattern-based video summarization technique: moving from low-level signals to high-level structure. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10 pp. vol.1–, Jan 2000.

[77] J. N. Ouellet and V. Randrianarisoa. To watch or not to watch: Video summarization with explicit duplicate elimination. In *Computer and Robot Vision (CRV), 2011 Canadian Conference on*, pages 340–346, May 2011.

[78] G. Dudek and J. P. Lobos. Towards navigation summaries: Automated production of a synopsis of a robot trajectories. In *Computer and Robot Vision, 2009. CRV '09. Canadian Conference on*, pages 93–100, May 2009.