



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EXTENSIÓN DE FILTRO DE KALMAN DE APROXIMACIÓN NO LINEAL PARA LA
DETECCIÓN DE OBJETOS ASTRONÓMICOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERA CIVIL EN COMPUTACIÓN

PALOMA CECILIA PÉREZ GARCÍA

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
AIDAN HOGAN

SANTIAGO DE CHILE
2019

RESUMEN

El presente trabajo describe el desarrollo de un software en PYTHON destinado a la detección de fenómenos astronómicos transitorios como las supernovas que corresponden a eventos caracterizados por un incremento rápido en su luminosidad y un consecuente decrecimiento lento. El programa se diseñó sobre la base de una rutina ya implementada la cual hace uso de estimaciones generadas por métodos del filtro de Kalman: en su versión clásica (o básica) o su versión de máxima correntropía. Debido a que esta rutina presenta complicaciones en la administración de archivos y manejo de parámetros (producido principalmente por *hard-coding*) se realizó un proceso de *refactoring* que implica además diseñar y generar una nueva familia de filtros de Kalman basados en el patrón de diseño STRATEGY.

Sobre este código refactorizado se efectuaron pruebas de rendimiento obteniéndose así una mejora en términos de tiempo pero no en la memoria principal utilizada. Por otro lado se realizaron pruebas de detección usando el conjunto de 93 supernovas detectadas por el sondeo de HiTS del año 2015, hallándose mejoras notables en la disminución de falsos positivos así como también un leve aumento en el número de verdaderos positivos al emplear las versiones clásica y de máxima correntropía de los filtros refactorizados. Sin embargo no ocurrió lo mismo con el nuevo filtro unscented, que permite emplear funciones no lineales al momento de estimar. Para este filtro se usaron una función cuadrática y otra de exponente 1,5; evaluadas sobre el paso del tiempo desde el inicio de las observaciones (o épocas).

Se recomienda continuar estudiando el nuevo filtro de Kalman de aproximación no lineal debido al acotado conjunto de parámetros y funciones utilizado durante la realización de este trabajo.

En memoria de mi padre

Agradecimientos

Para empezar, quiero agradecer a todos quienes participaron en el desarrollo de este trabajo; en particular agradezco enormemente el apoyo y la guía del profesor Pablo Estévez, con quién pude aprender aplicaciones desde su disciplina, y del profesor Benjamín Bustos por el tiempo y paciencia brindados durante el desarrollo de este trabajo de título. También agradezco a Francisco Förster por su orientación y tiempo entregados en los momentos de consulta. De igual modo, le doy las gracias al Laboratorio Nacional de Computación de Alto Rendimiento por las herramientas facilitadas, sin las cuales este proyecto no hubiese podido llevarse a cabo.

Quiero agradecer a toda la gente que he conocido en la U, tanto en mi licenciatura de astronomía como en la carrera de ingeniería del DCC, ya que siento que he aprendido de todos un poco, tanto de profesores como de compañeros y funcionarios.

Agradezco por contar con mi familia y amigos. A mi madre y mi hermana en especial, por acompañarme siempre en todo momento, aunque sea a la distancia.

Paloma Cecilia Pérez García

Tabla de Contenido

1. Introducción	13
1.1. Motivación	13
1.2. Objetivos	14
1.2.1. Objetivo general	14
1.2.2. Objetivos específicos	14
1.3. Organización de la tesis	15
2. Antecedentes	16
2.1. Supernova tipo II	16
2.2. High Cadence Transient Survey: HiTS	17
2.2.1. Datos obtenidos durante el año 2015	19
2.3. El filtro de Kalman	20
2.3.1. Filtro de Kalman Básico	23
2.3.2. Filtro de Kalman de Máxima Correntropía	27
2.3.3. Filtro de Kalman Unscented (UKF)	31
2.4. Laboratorio Nacional de Computación de Alto Desempeño (NLHPC)	35
3. Evaluación del programa original	36
3.1. El programa	36
3.2. Estructura de datos	38
3.3. Pruebas	38
3.3.1. Tiempo de ejecución	40
3.3.2. Uso de memoria	42
3.3.3. Detección	44
3.3.4. Observaciones	45
4. Refactoring	48
4.1. Manejo de la rutina: ROUTINEHANDLER	48

4.2. Manejo de datos de entrada	49
4.3. Determinación de flujos	50
4.4. Filtros originales	50
4.4.1. Predicción	51
4.4.2. Corrección	51
4.4.3. Filtros refactorizados	52
4.5. Detección de candidatos	52
4.6. Visualización de resultados	53
5. Filtro de aproximación no lineal: unscented Kalman	60
5.1. DISEÑO	60
5.1.1. Predicción	61
5.1.2. Corrección	62
6. Resultados	64
6.1. Desempeño	66
6.1.1. Tiempo de ejecución	66
6.1.2. Uso de memoria	68
6.2. Detección	70
6.2.1. Filtros básico y de máxima correntropía	70
6.2.2. Filtro unscented	72
6.3. Observaciones	72
7. Conclusión	76
7.1. Trabajo futuro	77
Glosario	79
Bibliografía	81
Apéndices	83
A . Librerías usadas para el refactoring	83
B . Rutas a directorios y expresiones regulares de archivos	84
B .1. Archivo de entrada: configuración de paths	85
C . Diccionario de parámetros y umbrales	86
C .1. Archivo de entrada: parámetro y umbrales	87
D . Archivo de entrada: lista de campos, CCDs y semestres (incluye algunas coordenadas)	88
E . Métodos de la clase ROUTINEHANDLER	89

F . Métodos de la clase DATAPICKER	90
G . Métodos de la clase SOURCEFINDER	92
H . Métodos en utils	94
I . Métodos en unscented_utils	95
J . Detección usando pipeline original	96
K . Detección usando pipeline refactorizada	97
L . Detección usando filtro unscented	98
M . Diagrama de clases de programa original	99
N . Diagrama de clases del programa refactorizado	100

Índice de tablas

3.1. Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación del filtro, agrupación de píxeles y filtrado de los mismos durante el período de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman básico. La última fila corresponde a la media por observación.	41
3.2. Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación de los filtros, agrupación de píxeles y filtrado de los mismos durante el período de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman básico. Se observa que para las dos últimas supernovas los tiempos son cero ya que no se encontraron más candidatos.	41
3.3. Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación del filtro de Kalman, agrupación de píxeles y filtrado de los mismos durante el período de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman de máxima correntropía.	41
3.4. Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación de los filtros, agrupación de píxeles y filtrado de los mismos durante el período de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman de máxima correntropía. Se observa que para las dos últimas supernovas los tiempos son cero ya que no se encontraron más candidatos.	42
3.5. Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman básico. La última fila corresponde a tiempo total promedio por observación.	42
3.6. Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman de máxima correntropía.	42

3.7. Memoria principal (en unidades de MB) usada durante la ejecución del programa original con la versión básica del filtro de Kalman.	43
3.8. Memoria principal (en unidades de MB) empleada durante la ejecución del programa original usando filtro de Kalman de máxima correntropía.	44
3.9. Número de verdaderos positivos (TP), falsos negativos (FN) y falsos positivos (FP) (objetos no considerados por HiTS como supernova) encontrados usando cada uno de los filtros. La cuarta columna, NaN indica el número conjunto de datos que no se pudieron procesar por falta de algún(os) archivo(s). No se observan diferencias sustanciales entre los resultados de cada filtro. Con el filtro de máxima correntropía se encontró un falso positivo menos.	45
6.1. Tiempo de ejecución en segundos de cada proceso involucrado, usando el filtro de Kalman básico refactorizado: cálculo de flujo, estimación de estados, detección de candidatos y guardado de candidatos resultantes en caso de haberlos. La última fila describe el tiempo promedio que toma por observación (en segundos igualmente) para cada uno de los procesos.	67
6.2. Tiempo de ejecución en segundos de cada proceso involucrado, usando el filtro de Kalman de máxima correntropía refactorizado: cálculo de flujo de las imágenes, estimación de estado, detección de fuentes y guardado de candidatos. La última fila describe el tiempo promedio que toma por observación (en segundos) para cada una de las tareas.	67
6.3. Tiempo de ejecución en segundos de cada tarea, usando el filtro de Kalman unscented: cálculo de flujo, estimación de filtros, detección de fuentes y guardado de candidatos. La última fila describe el tiempo promedio que toma por observación (en segundos) para cada uno de los procesos.	67
6.4. Tiempo de exploración (para detección de candidatos) para cada uno de los filtros. La última fila corresponde a tiempo (en segundos) total promedio por observación. A la izquierda, tiempo empleado usando la nueva versión de filtro de Kalman básico. En el centro, los resultados de tiempo correspondientes al filtro de Kalman de máxima correntropía. A la derecha, resultados correspondientes al filtro unscented.	68

6.5. Comparación de la memoria principal (en unidades de MB) usada durante la ejecución del programa para los tres filtros. A la izquierda, los resultados correspondientes al usar filtro de Kalman Básico. En el centro, memoria ocupada usando el filtro de Kalman de máxima correntropía. A la derecha, memoria ocupada usando filtro de Kalman unscented.	69
6.6. Número de verdaderos positivos (TP), falsos negativos (FN) y falsos positivos (FP) encontrados usando cada uno de los filtros en el conjunto de datos de HiTS. La cuarta columna de valores muestra la cantidad de conjuntos de datos que no pudieron ser procesados.	71
6.7. Número de verdaderos positivos (TP), falsos negativos (FN) y falsos positivos (FP) encontrados usando el filtro unscented para dos tipos de funciones no lineales: $f(\Delta t) = (\Delta t)^{1.5}$ y $f(\Delta t) = (\Delta t)^2$, sobre el dataset de HiTS. La tercera columna de valores muestra la cantidad de conjuntos de datos que no pudieron ser procesados.	72
7.1. Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando los filtros implementados originalmente (básico y de correntropía máxima).	96
7.2. Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando los filtros refactorizados (básico y de correntropía máxima).	97
7.3. Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando el filtro unscented para una función $f(\Delta t) = \Delta t^{1.5}$ y $f(\Delta t) = \Delta t^2$	98

Índice de figuras

- 2.1. Esquema no a escala de la estructura de una estrella masiva previo a su explosión como supernova. Los elementos más pesados se alojan en el centro, mientras que los más livianos, como el hidrógeno o el helio lo hacen en la capa más externa. *Imagen publicada por R. J. Hall en Wikimedia Commons, 15 de agosto de 2007.* 17
- 2.2. Curvas típicas de supernovas Ia (enana blanca) y II (estrella masiva). ©2004 Pearson Education Inc., publishing as Addison-Wesley *The Bizarre Stellar Graveyard.* 18
- 2.3. Diagrama que ilustra la evolución del brillo de una supernova en términos de luminosidad solar (L_{\odot}) durante días. Se resalta el fenómeno de *shock-breakout* apenas comienza el incremento de la luminosidad de la supernova. Esta imagen fue publicada en la página de la NASA destacando la primera vez que un evento como este es *capturado* en la banda visible (por el telescopio espacial Kepler). *NASA Ames/W. Stenzel, 2016.* 18
- 2.4. A la izquierda, estructura de la cámara DECam poblada con 62 chips CCDs. A la derecha, imagen *flat field* desde DECam. *Imágenes tomadas desde la página del Dark Energy Survey (www.darkenergysurvey.org/the-project/instrument/the-camera).* 19
- 2.5. Orientaciones sobre el cielo y la huella espacial del arreglo de detectores en el plano focal. Se destacan los CCD cuya etiqueta comienzan con S o N, ya que estos corresponden a los detectores encargados de obtener las imágenes científicas. El etiquetado de estas componentes puede ser engañoso debido a que las iniciales de norte (North) y sur (South) están invertidas en relación a la orientación del cielo. *Imagen publicada en el sitio del CTIO (<http://www.ctio.noao.edu/noao/node/2250>).* 20
- 2.6. Distribución espacial de los campos observados durante los primeros semestres de los años 2013 (gris), 2014 (azul) y 2015 (naranja). En tono rojo, los mismos campos del año 2015 y 2014 (superposición). *F. Förster et al., 2015. HiTS real-time supernova detections.* 21

2.7. Frecuencia de las observaciones realizadas en el survey de HiTS durante el semestre del año 2015.	21
2.8. Diagrama del proceso de estimación de estados. El filtro de Kalman “sigue la pista” del estado físico de un sistema, estimándolo paso a paso. El proceso comienza con la entrada de los valores $\hat{x}_{k-1 k-1}$ y $P_{k-1 k-1}$, determinados en el instante $k - 1$. Para obtener la estimación del estado en el instante actual, k , se hace una predicción a partir de la información más reciente (obtenida en $k - 1$) y del modelo físico escogido, obteniéndose así los valores de $\hat{x}_{k k-1}$ y $P_{k k-1}$. Luego durante la fase corrección esta predicción es corregida con la observación z_k y con ello, se actualizan los valores del estado predicho y de la matriz de covarianza en $\hat{x}_{k k}$ y $P_{k k}$, respectivamente. Finalmente estas últimas cantidades se transformarán en la entrada del algoritmo para el tiempo $k + 1$. <i>Imagen publicada por P. Aimonen en Wikimedia Commons, en noviembre de 2011.</i>	23
2.9. Representación del proceso de predicción (obtención de cantidades a priori) de las cantidades $\hat{x}_{k k-1}$ y $P_{k k-1}$; y de corrección (estimación a posteriori) para obtener las cantidades $\hat{x}_{k k}$ y $P_{k k}$. Para el siguiente paso, $k + 1$, estas estimaciones pasan a ser entrada (<i>input</i>) de un nuevo proceso de predicción: $\hat{x}_{k+1 k}$ y $P_{k+1 k}$. <i>E. Matsinos, 2016. The Kalman Filter: a didactical overview.</i>	25
2.10. Representación del funcionamiento del filtro UKF. En esta oportunidad se hace uso de la función f y h para obtener las transformaciones $x_{k-1} \rightarrow x_k$ y $x_k \rightarrow z_k$. Esto se logra con la evaluación de los $2N+1$ <i>puntos sigma</i> generados durante la etapa de predicción (y posteriormente en la etapa de corrección). <i>E. Matsinos, 2016. The Kalman Filter: a didactical overview.</i>	32
3.1. La figura de la izquierda, corresponde a una muestra de una imagen científica, en donde se logra observar sus píxeles los cuales deben ser evaluados individualmente primero. Luego, en la figura derecha, se destaca en un círculo rojo, la supernova conocida (SN34) y en círculos verdes las fuentes candidatas a supernova, como resultado del proceso de agrupamiento de los píxeles que satisfacen cierto conjunto de criterios.	37

3.2. Diagrama de flujo del programa original. Se aprecian dos ciclos principales: el primero est <acute>a destinado a la b<acute>usqueda de una supernova en una coordenada dada y la revisi<acute>on de nuevos candidatos; y el segundo a la revisi<acute>on de la lista de candidatos encontrados en el primer bloque del proceso, para el guardado de la informaci<acute>on (extracci<acute>on de estampillas, desde cada matriz, centradas en sus coordenadas) de estos candidatos. Notar que hay pasos que se repiten en la realizaci<acute>on de ambos an<acute>alisis.</acute></acute></acute></acute></acute></acute></acute></acute>	39
3.3. Esquema de las estructuras de datos usadas para la representaci <acute>on de las matrices de estados y de las matrices de covarianza de la relaci<acute>on entre el flujo (x) y la velocidad de flujo (\dot{x}) por cada p<acute>ixel, de una imagen de ancho w y altura h. Para los conjuntos de datos, $w = 2046$ y $h = 4094$.</acute></acute></acute>	40
3.4. Comportamiento de la memoria (en mebibytes) durante la ejecuci <acute>on para los tres conjuntos de datos de las supernovas SN14, SN18 y SN80, usando el filtro de Kalman b<acute>asico.</acute></acute>	43
3.5. Comportamiento de la memoria (en mebibytes) durante la ejecuci <acute>on para los tres conjuntos de datos de las supernovas SN14, SN18 y SN80. En los tres lanzamientos se us<acute>o el filtro de Kalman de m<acute>axima correntrop<acute>ia.</acute></acute></acute></acute>	44
3.6. Curvas de luz (flujo en ADU vs tiempo en MJD) de 16 supernovas detectadas tanto por el filtro de Kalman b <acute>asico como el de m<acute>axima correntrop<acute>ia. Esta figura corresponde a un an<acute>alisis exploratorio de resultados obtenidos a trav<acute>es de m<acute>etodos fotom<acute>etricos tradicionales (no por filtros de Kalman), y est<acute>an disponibles junto a la base de datos de HiTS [8].</acute></acute></acute></acute></acute></acute></acute></acute>	45
3.7. Curvas de luz (flujo en ADU vs tiempo en MJD) de 16 supernovas no detectadas ni por el filtro de Kalman b <acute>asico ni por el de de m<acute>axima correntrop<acute>ia. Esta figura corresponde a un an<acute>alisis exploratorio de resultados obtenidos a trav<acute>es de m<acute>etodos fotom<acute>etricos tradicionales (no por filtros de Kalman), y est<acute>an disponibles junto a la base de datos de HiTS [8]. Se aprecia el r<acute>apido crecimiento en la luminosidad de una de ellas.</acute></acute></acute></acute></acute></acute></acute></acute></acute>	46
3.8. Curva de luz (en ADU vs MJD) de supernova registrada en el CCD N27, campo 34. La detecci <acute>on realizada por los filtros b<acute>asico y de m<acute>axima correntrop<acute>ia se realiz<acute>o en el MJD 57072.214 (época u observaci<acute>on 7 de 18).</acute></acute></acute></acute></acute></acute>	47
3.9. Curva de luz supernova observada en el CCD S5, campo 21. Ambos filtros detectaron la supernova en el MJD 57077.166 (época u observaci <acute>on 23 de 27).</acute>	47

4.1. Rutina del programa refactorizado. Cuenta con dos modos de ejecución: uno de búsqueda de candidatos (izquierda) y otro de generación de gráficos (derecha). Pueden ejecutarse ambas formas comenzando con la búsqueda de candidatos y continuar con el modo de generación de gráficos. El usuario tiene la libertad de escoger el modo de ejecución.	49
4.2. Familia de filtros de Kalman y patrón <i>strategy</i> usado en la implementación de los métodos <code>predict</code> y <code>correct</code>	52
4.3. Conjunto de series de tiempo de diferentes componentes de interés en el pixel ubicado en la coordenada de posición del candidato. El primer gráfico (de arriba hacia abajo) muestra la evolución del flujo medido en contraste con el comportamiento de la predicción y estimación realizada por el filtro de Kalman del mismo flujo durante las épocas de las observaciones. La segunda gráfica muestra los cambios de la predicción y estimación de la velocidad de flujo (obtenidas por el filtro de Kalman) en el tiempo. El tercer esquema muestra la evolución de las etiquetas (grupal e individual) del pixel ubicado en la coordenada del candidato. Finalmente, el último gráfico visualiza el comportamiento de las diferentes varianzas y covarianzas tanto de las componentes predichas y estimadas por el filtro (flujo y velocidad de flujo), así como de las mediciones del mismo flujo (observado).	57
4.4. Estampillas de matrices de 21×21 píxeles y etiquetas que describen su comportamiento a través del tiempo: la primera fila de imágenes corresponde a estampillas obtenidas desde las imágenes científicas en donde debiese habitar la supernova observada durante todas las observaciones. La segunda fila muestra los diferentes modelos de PSF obtenidos para diferentes épocas. La tercera fila muestra el flujo observado en la misma posición. Le sigue la varianza de este flujo. Posteriormente viene el flujo estimado por el filtro de Kalman siguiendo la velocidad de flujo estimado. Luego vienen las etiquetas de los píxeles reconocidos por el programa como pertenecientes a un objeto transitorio (etiquetado por pixel y por grupo de píxeles). La última fila corresponde a la máscara base usada durante el análisis.	58
4.5. Espacio de fase de flujo y velocidad de flujo de un candidato. En azul se destaca la estimación lograda por filtro de Kalman y en rojo el flujo observado versus la misma estimación de velocidad. Notar que en la leyenda de la figura, se indica el nivel de complejidad de la curva estimada en términos de su entropía [3] para la curva de flujo y velocidad de flujo estimado.	59

5.1. Diagrama de clases de la familia de filtros de Kalman resultante. El patrón Strategy permite que las funcionalidades de predicción y corrección puedan definirse en diferentes clases, facilitando la modularización y la implementación de nuevos filtros como el unscented.	63
6.1. Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos usando el filtro de Kalman Básico.	69
6.2. Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman de máxima correntropía.	70
6.3. Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman unscented. .	71
6.4. En la primera gráfica de arriba hacia abajo se muestra el comportamiento del flujo medido y el flujo predicho y estimado. En esta representación se observa que el flujo se dispara a partir de la fecha 57075. De la misma forma, se dispara las velocidades de flujo estimado y predicho de acuerdo al segundo gráfico. El siguiente esquema indica el crecimiento de <i>flags</i> de invalidez de los píxeles mientras que la última imagen da cuenta del rápido crecimiento de las varianzas y covarianzas de estimación y predicción de las variables de estado.	73
6.5. Representación en píxeles de lo que sucede con el flujo observado, las estimaciones y los <i>flags</i> de los píxeles. Se distingue la saturación sobre las matrices de flujo y velocidad estimada.	74
6.6. Se observa el comportamiento conjunto del flujo y velocidad estimados. El rápido crecimiento de ambas variables es bastante notable	74
7.1. Mientras más cerca esté el objeto del horizonte, mayor será la cantidad de airmass por la que se observará y mayor será la distorsión sobre la imagen percibida del objeto. <i>Reed, C. 2008; https://www.skyandtelescope.com/astronomy-resources/transparency-and-atmospheric-extinction/</i>	79
7.2. Ejemplo de distorsión de una fuente al aplicar un kernel de PSF específico. El resultado se observa en el cuadro <i>Image. P. Huentelemu, 2016 [15]</i>	80
7.3. Diagrama de clases del programa original. Se listan cada uno de métodos y variables de cada clase.	99

1 | Introducción

1.1. Motivación

La astronomía es uno de los campos científicos que más se ha visto afectado por el rápido crecimiento en la generación de datos debido al fuerte desarrollo de nuevas tecnologías de la información y de nuevos instrumentos destinados a la observación. Este crecimiento ha gatillado un aumento importante en la demanda de una nueva generación de métodos que puedan procesar esta oleada de información o big data astronómico.

Ejemplos de proyectos que actualmente producen una gran cantidad de datos a través de telescopios en diferentes partes del mundo son: el Panoramic Survey Telescope and Rapid Response System (Pan-STARRS) [10], el Visible and Infrared Survey Telescope (VISTA) [7], el VLT Survey Telescope (VST) [2], el Dark Energy Camera Legacy Survey (DECaLS) [4] y el Hyper Suprime-Cam Subaru Strategic Program (HSC SSP) [1]. Estos surveys están caracterizados por un amplio *étendue* (extensión óptica) definido como el producto entre el área del espejo de un telescopio y su ángulo sólido proyectado en el cielo.

En el futuro, telescopios como el Large Synoptic Survey Telescope (LSST) [12] (que entrará en funcionamiento a mediados del 2022) continuarán revolucionando la era del big data en astronomía con étendues y cámaras CCD mucho más grandes de lo que se utiliza hasta el día de hoy. En particular se espera que el LSST produzca un número de alertas de fenómenos transitorios (avisos de objetos que cambian en el tiempo o espacio) del orden de 10 millones cada noche. La capacidad de detectar nuevos objetos de interés dependerá de la calidad de los datos y de los algoritmos de tiempo real destinados a generar las alertas mencionadas.

Al día de hoy se han elaborado sondeos como el High Cadence Transient Survey (HiTS) [8] cuya finalidad ha sido la búsqueda de fenómenos transitorios rápidos con escalas de tiempo que van desde las horas a días, utilizando secuencias de observaciones de la cámara DECam del telescopio Blanco (en Cerro Tololo) para la detección y posterior reporte de objetos candidatos a supernova.

En este trabajo se propone un método de detección de potenciales candidatos a supernova a través de la discriminación de píxeles que involucren un incremento en la intensidad. Esta discriminación comienza con la determinación del flujo a través de la intensidad de los píxeles que puedan corresponder a una estrella y a la variación de cada uno de ellos usando métodos iterativos de filtrado en secuencias de imágenes de largo arbitrario. Los filtros desarrollados para HiTS corresponden a miembros de una familia de filtros conocidos como *filtros de Kalman*.

El trabajo desarrollado por Pablo Huentelemu [15] (M.Sc.), propone el uso de los filtros clásico (o básico) [11] y de máxima correntropía [6] en el reconocimiento de supernovas jóvenes (en su fase de crecimiento en su luminosidad) de tipo II, pero que a pesar de acertar en algunas detecciones en el conjunto de datos de HiTS de supernovas confirmadas, muchas no son reconocidas por lo que se busca mejorar su resultado. De modo que se plantea la posibilidad de diseñar algún otro criterio de filtrado y de estudiar la variación de los resultados.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo de esta tesis comprende la reestructuración y extensión de un programa existente destinado al análisis fotométrico de datos astronómicos con el cual se busca implementar un modelo de sistema de alertas que dé aviso del estado temprano de un fenómeno astronómico transitorio.

1.2.2. Objetivos específicos

- Mejorar proceso existente sobre la manipulación de los archivos requisito del programa original, debido a que actualmente existen problemas de *hard-coding* respecto de la ubicación de estos, así como ciertos reparos durante el proceso de selección de los mismos. Además se necesita reformular la configuración de parámetros funcionales de entrada (propios de la ejecución del programa, como los umbrales) para flexibilizar su entrada.
- Generar una familia de métodos en términos de programación orientada a objetos empleando el patrón de diseño *Strategy* que implemente la familia de métodos conocida como Filtros de Kalman.
- Agregar una nueva variante de filtro que permita el uso de un modelo no lineal.

- Implementar gráficas de resultados en las que sea posible observar la curva generada por el flujo y la velocidad de flujo estimados por los filtros, indicando la entropía aproximada de la curva en el espacio de estados y visualizar la evolución de las mediciones y estimaciones tanto en secuencias de imágenes estampillas (píxeles) como en gráficas (series de tiempo).
- Estudiar los resultados obtenidos con esta nueva versión del programa con cada uno de sus filtros sobre un conjunto de datos brindado por HiTS (del año 2015), a la par que el desempeño (medido en términos de uso de memoria principal y tiempo de ejecución) de éste. Por último se busca establecer un contraste con la versión original.

1.3. Organización de la tesis

Este documento contiene los siguientes capítulos: en el Capítulo 2 se describe en qué consiste una supernova, el proyecto HiTS, el background matemático de los métodos de filtrado usados en el trabajo original (filtros de Kalman básico y de máxima correntropía) y el filtro a implementar (filtro de Kalman unscented).

En el Capítulo 3, se describen aspectos generales del programa original y se estudian los resultados de desempeño obtenidos con él, estableciéndose una breve discusión.

Posteriormente, en el Capítulo 4 se describe el refactoring del programa del código original y se detalla la construcción de la familia de métodos de filtros de Kalman usando el patrón Strategy y del rediseño de las funciones de visualización, incluyendo la generación de la curva de estados.

Luego, en el Capítulo 5 se detalla la implementación del nuevo filtro de aproximación no lineal.

A continuación, en el Capítulo 6, se exponen los resultados obtenidos para el desempeño de la nueva versión del programa (para los dos filtros originales) incluyendo la cantidad de supernovas redescubiertas y el período en que se detectaron. El capítulo concluye con el análisis de los resultados y una comparación con aquellos obtenidos en el Capítulo 3.

Finalmente en el Capítulo 7 se presentan las conclusiones de este trabajo y se propone el trabajo a futuro a realizar.

2 | Antecedentes

En el presente Capítulo se exponen conceptos cruciales para la comprensión y desarrollo de este trabajo. Se describen en particular los objetos de interés que estimularon el desarrollo del survey HiTS: las supernovas de tipo II y el evento de shock-breakout asociado a éstas y una breve descripción de los datos obtenidos por este proyecto durante el año 2015. De la misma forma, se describe la base matemática en la que se basan las diferentes versiones del filtro de Kalman implementadas en el software original, es decir, los filtros de Kalman *básico* y de *máxima correntropía* así como una nueva versión del método conocida como *unscented* que permitiría la introducción de modelos no lineales en el proceso de filtrado.

2.1. Supernova tipo II

Una supernova de tipo II corresponde a un evento estelar con el que finaliza la vida de una estrella masiva (aquellas que en su proceso de formación poseen una masa superior a 10 masas solares ¹). Al no contar con combustible necesario para llevar a cabo reacciones nucleares que puedan contrarrestar su propia gravedad (su núcleo ya no puede formar elementos más pesados que el hierro o el níquel, los cuales caen al núcleo de la estrella por su propio peso. Ver Figura 2.1.), y si la presión degenerada² de los electrones del plasma de la estrella no es suficiente para soportar este peso, la estrella se contrae abruptamente incrementando la temperatura de su centro a, aproximadamente, 10^{10} K.

Debido al aumento de temperatura, los electrones del núcleo adquieren energía cinética suficiente para escapar, desapareciendo así la presión que ejercían hacia el exterior. Finalmente el núcleo colapsa liberando energía gravitacional y con ella las capas más externas de la estrella son expulsadas en una gran explosión. Este fenómeno se denomina *supernova* e implica un aumento repentino del brillo de una estrella, incrementando su brillo en un factor de 10^8 veces, pudiendo incluso ser más brillante que la galaxia que la alberga.

¹ $M_{\odot} = (1.98847 \pm 0.00007) \times 10^{30}$ Kg

²presión que viene del principio de exclusión de Pauli

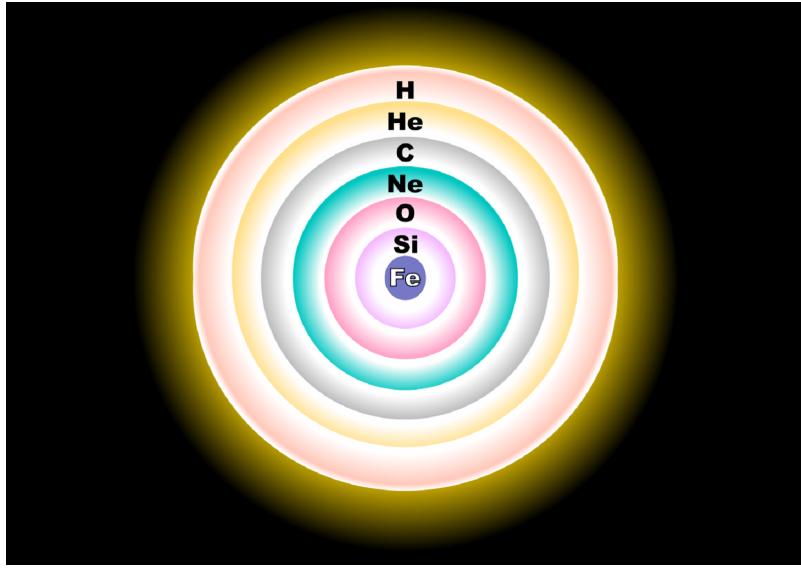


Figura 2.1: Esquema no a escala de la estructura de una estrella masiva previo a su explosión como supernova. Los elementos más pesados se alojan en el centro, mientras que los más livianos, como el hidrógeno o el helio lo hacen en la capa más externa. *Imagen publicada por R. J. Hall en Wikimedia Commons, 15 de agosto de 2007.*

En general la variación de la luminosidad de una supernova corresponde a una curva que crece rápidamente los primeros días (u horas), alcanzando un máximo, para luego decaer. Cabe destacar que existen otros tipos de supernova, como las de tipo Ia que corresponden a otro fenómeno en donde participa una clase de estrella denominada enana blanca junto a otra estrella de cualquier otro tipo. La primera, al poseer una gravedad tan alta en su superficie es capaz de tomar material de su compañera con lo que al superar las $1.44 M_{\odot}$ se desencadenaría la explosión de una supernova.

Las supernovas de tipo Ia presentan un decaimiento casi continuo una vez alcanzado el máximo, mientras que las de tipo II presentan dos caídas: una inmediatamente después de su máximo y otra una vez finalizado un período de decaimiento suavizado (ver Figura 2.2). Otra forma de diferenciarlas es la presencia de trazas de hidrógeno en los espectros de éstas: las supernovas de tipo Ia prácticamente no presentan hidrógeno (líneas de absorción distintivas) a diferencia de las de tipo II.

2.2. High Cadence Transient Survey: HiTS

El High Cadence Transient Survey [8] (desde ahora, HiTS) es un survey cuyo objetivo principal es detectar y seguir fenómenos transitorios estelares en escalas de tiempo que van desde horas a días, con especial atención a fases tempranas de explosiones de supernovas (primeras horas). Sin

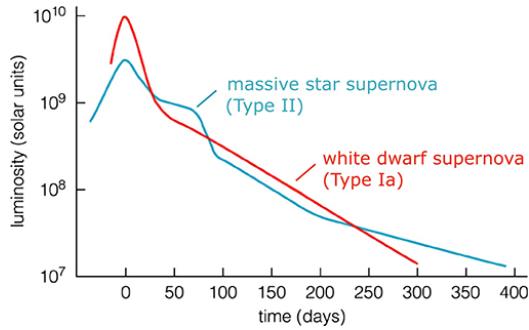


Figura 2.2: Curvas típicas de supernovas Ia (enana blanca) y II (estrella masiva). ©2004 Pearson Education Inc., publishing as Addison-Wesley *The Bizarre Stellar Graveyard*.

embargo, el objetivo original de HiTS corresponde a la detección de un fenómeno llamado *shock breakout* (SBO), un fenómeno que ocurre inmediatamente después del colapso del núcleo de una estrella roja supergigante (una de las posibles etapas finales de una estrella masiva antes de *explorar* en supernova II). Ver Figura 2.3³.

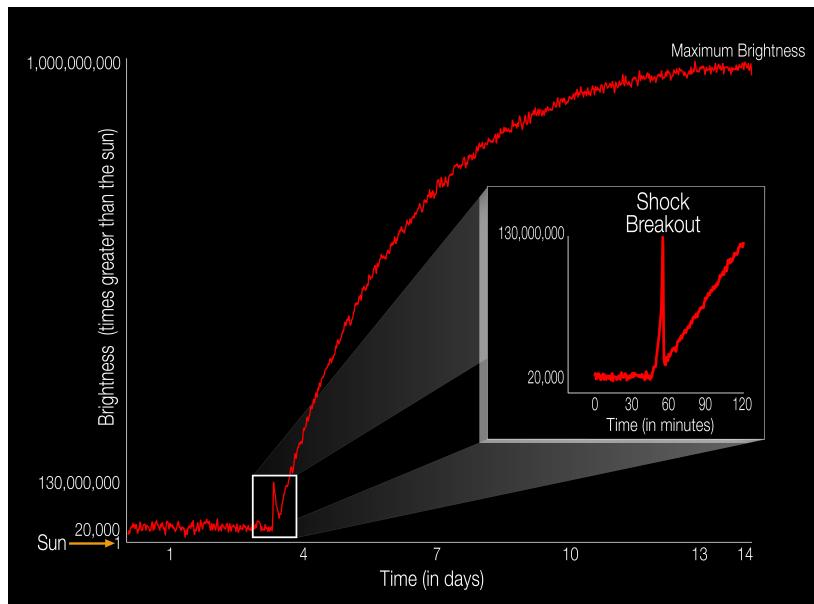


Figura 2.3: Diagrama que ilustra la evolución del brillo de una supernova en términos de luminosidad solar (L_{\odot}) durante días. Se resalta el fenómeno de *shock-breakout* apenas comienza el incremento de la luminosidad de la supernova. Esta imagen fue publicada en la página de la NASA destacando la primera vez que un evento como este es *capturado* en la banda visible (por el telescopio espacial Kepler). *NASA Ames/W. Stenzel, 2016*.

HiTS utiliza la Dark Energy Camera (DECam, ver Figura 2.4) para la obtención de sus imágenes. Esta cámara se encuentra montada en el Telescopio Blanco del Observatorio de Cerro Tololo (CTIO) en la región de Coquimbo, Chile. Esta cámara posee 62 detectores CCD de 2048×4096 píxeles para la obtención de imágenes científicas y otros 12 para la guía, alineamiento y enfoque (ver Figuras 2.4

³ $L_{\odot} = 3.828 \times 10^{26}$ W

y 2.5 dónde se muestra disposición de las cámaras CCD en el telescopio).

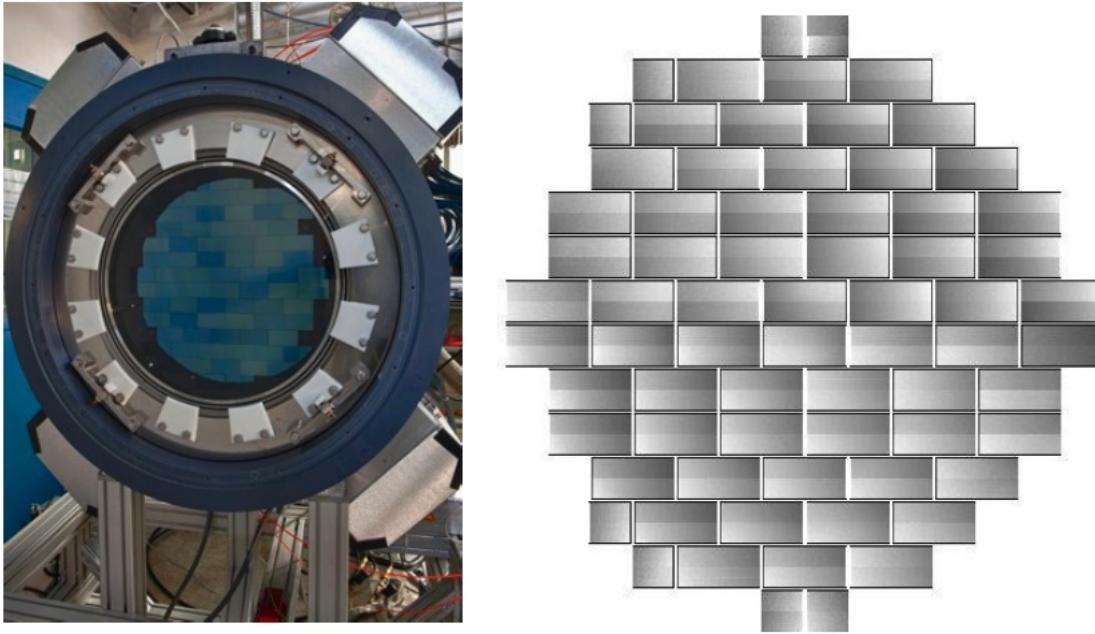


Figura 2.4: A la izquierda, estructura de la cámara DECam poblada con 62 chips CCDs. A la derecha, imagen *flat field* desde DECam. *Imágenes tomadas desde la página del Dark Energy Survey (www.darkenergysurvey.org/the-project/instrument/the-camera).*

Durante el proyecto HiTS se realizaron tres campañas de observación en los años 2013, 2014 y 2015 durante el primer semestre de cada año. Se escogieron 40 campos y 4 épocas por noche (también por campo) para observaciones realizadas en el 2013 y el 2014. Para la campaña del año 2015 se escogieron 50 campos. En estas campañas se obtuvieron más de 120 candidatos a supernova. Sin embargo, no se logró encontrar en estas rastros de SBO, evento que comprendió uno de los principales objetivos de HiTS. La distribución de los campos observados, por año, se describe en la Figura 2.6.

2.2.1. Datos obtenidos durante el año 2015

Como se mencionó anteriormente, la campaña del año 2015 tuvo lugar el primer semestre de ese año. El período en que se llevó a cabo fue durante los meses de febrero y marzo; específicamente entre los días 17 de febrero y 14 de marzo.

El período comprendido por los días 17 a 22 de febrero fue el de mayor latencia, obteniéndose hasta cinco épocas (observaciones) por noche por cada campo. Posteriormente la toma de observaciones cesa y se reanuda a partir del 25 de febrero con una latencia de a lo más tres épocas por noche y campo, con interrupciones de hasta nueve días finalizando el 14 de marzo (ver Figura 2.7).

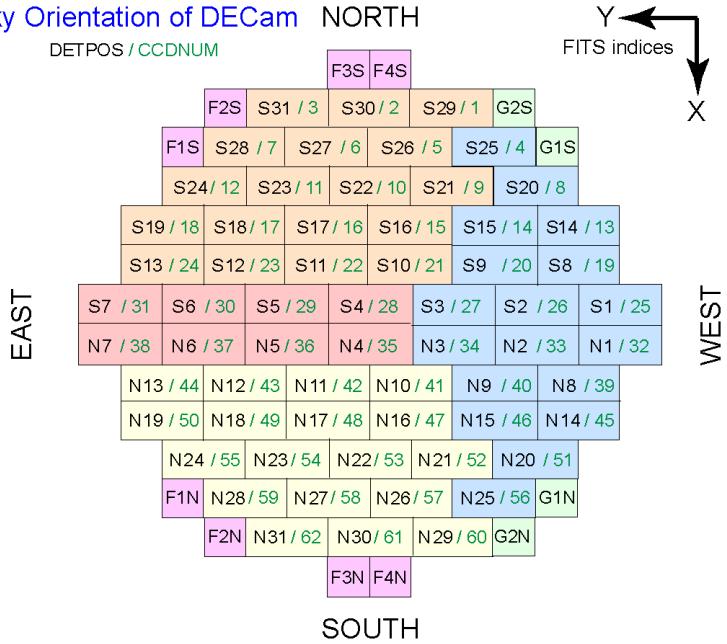


Figura 2.5: Orientaciones sobre el cielo y la huella espacial del arreglo de detectores en el plano focal. Se destacan los CCD cuya etiqueta comienzan con S o N, ya que estos corresponden a los detectores encargados de obtener las imágenes científicas. El etiquetado de estas componentes puede ser engañoso debido a que las iniciales de norte (North) y sur (South) están invertidas en relación a la orientación del cielo. *Imagen publicada en el sitio del CTIO (<http://www.ctio.noao.edu/noao/node/2250>).*

2.3. El filtro de Kalman

La evolución determinística de un sistema físico en el tiempo es conocida si el estado del sistema es medido con absoluta precisión en cada instante de tiempo (i.e., en un entorno donde es posible despreciar fenómenos cuánticos). Sin embargo toda medición está sujeta a incertezas finitas. Para sistemas que son observados en intervalos prolongados de tiempo, se prevee que las diferencias entre los estados estimados y los medidos se incrementen con el tiempo. Para la obtención de predicciones lo más confiables posible se requiere que el sistema sea regularmente monitoreado y sus estados estimados puedan ser considerados confiables en un lapso de tiempo apropiado.

Los filtros de Kalman son métodos que proveen un compromiso (o trade-off) entre los valores esperados del estado actual de un sistema y las mediciones que proporcionan información de su estado real. La aplicación de un filtro de Kalman está pensada como un proceso de dos fases:

1. **Fase predictiva:** Una *apuesta* del estado actual del sistema que se basa en un modelo físico (determinístico). Esta cantidad se denomina usualmente como estado estimado *a priori*.
2. **Fase correctiva:** La estimación del estado *a priori* es corregida con una medida real del

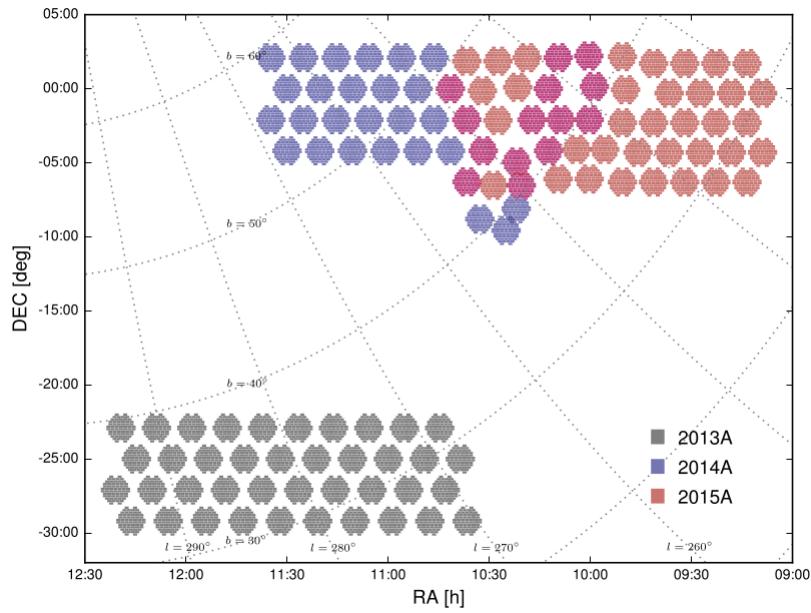


Figura 2.6: Distribución espacial de los campos observados durante los primeros semestres de los años 2013 (gris), 2014 (azul) y 2015 (naranjo). En tono rojo, los mismos campos del año 2015 y 2014 (superposición). *F. Förster et al., 2015. HiTS real-time supernova detections.*

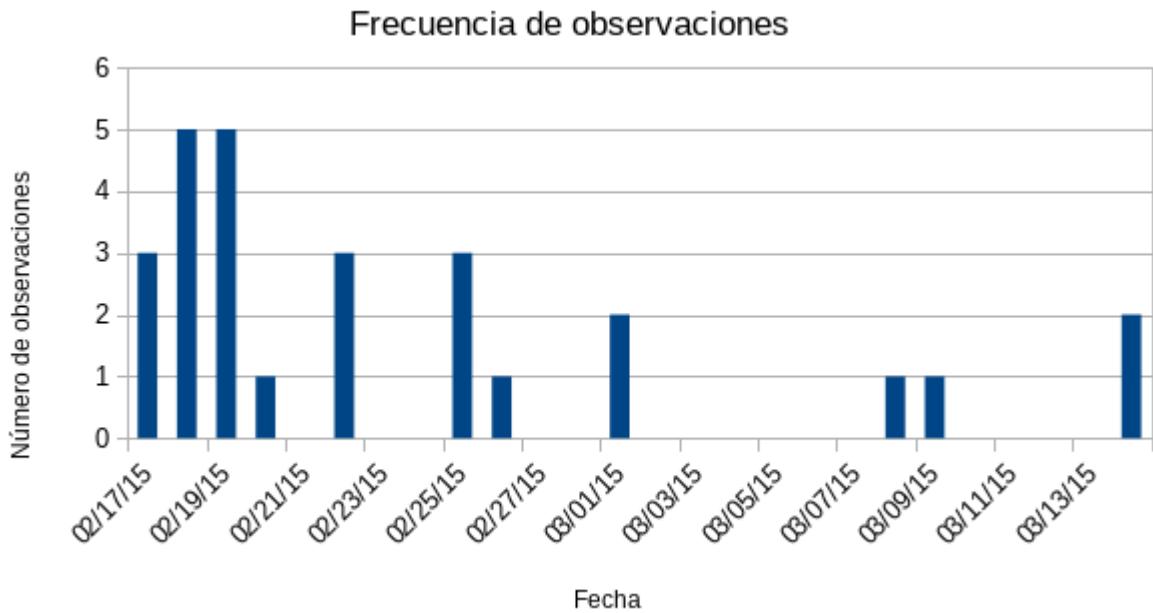


Figura 2.7: Frecuencia de las observaciones realizadas en el survey de HiTS durante el semestre del año 2015.

sistema con la que se obtiene la predicción; con ella se procede a calcular una cantidad conocida como *ganancia de Kalman* con la que se estima una *aproximación a posteriori* del sistema, evaluando que tan lejos estuvo nuestra aproximación *a priori*.

Típicamente estas fases de predicción y corrección se van alternando mientras se estudia el comportamiento físico de algún sistema. En particular, estos filtros son bastante usados en procesos como el guiamiento de un móvil, en sistemas de navegación y en análisis de señales; contextos en los cuales el monitoreo de estado de un sistema puede ser más que relevante.

En las subsecciones siguientes se hará uso de la notación de subíndices $m|n$, en las estimaciones de estado y covarianzas, para explicitar el instante de tiempo al cual pertenecen: m ; y al instante de tiempo de donde se extrae la información: n . Se hará uso de la notación k para referirse al estado actual y de $k - 1$ y $k + 1$ para indicar los instantes anterior y próximo a k , respectivamente.

- $\hat{x}_{k-1|k-1}$: Estado estimado en el paso anterior ($k - 1$) o estado inicial del modelo. Es un vector de largo N , donde N es el número de variables de estado a estudiar.
- $P_{k-1|k-1}$: Matriz de covarianza asociada al estado estimado en el paso anterior o matriz de covarianza inicial. Posee dimensión $N \times N$.

Durante la fase predictiva se obtienen las cantidades detalladas a continuación:

- $\hat{x}_{k|k-1}$: Estado estimado *a priori* (vector de dimensión N).
- $P_{k|k-1}$: Matriz de covarianza *a priori* (matriz de dimensión $N \times N$).

Luego, en la fase de corrección se recibe como entrada la medición realizada en el tiempo k , z_k , y se calculan las siguientes cantidades:

- \hat{z}_k : Combinación del estado estimado *a priori*, $\hat{x}_{k|k-1}$ y la medición real, z_k . Corresponde a la corrección, propiamente tal.
- \tilde{z}_k : Diferencia entre la medición real, z_k , y la corrección \hat{z}_k .
- $\hat{x}_{k|k}$: Estado estimado *a posteriori* o estado actualizado.
- $P_{k|k}$: Matriz de covarianza *a posteriori* (actualizada).

El proceso de estimación (predicción y corrección) puede resumirse en el diagrama de la Figura 2.8.

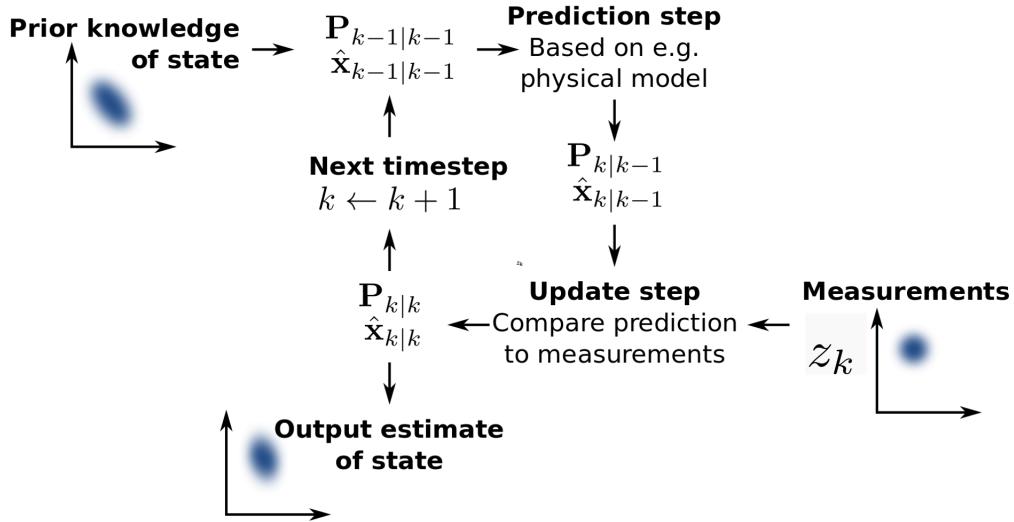


Figura 2.8: Diagrama del proceso de estimación de estados. El filtro de Kalman “sigue la pista” del estado físico de un sistema, estimándolo paso a paso. El proceso comienza con la entrada de los valores $\hat{x}_{k-1|k-1}$ y $P_{k-1|k-1}$, determinados en el instante $k-1$. Para obtener la estimación del estado en el instante actual, k , se hace una predicción a partir de la información más reciente (obtenida en $k-1$) y del modelo físico escogido, obteniéndose así los valores de $\hat{x}_{k|k-1}$ y $P_{k|k-1}$. Luego durante la fase corrección esta predicción es corregida con la observación z_k y con ello, se actualizan los valores del estado predicho y de la matriz de covarianza en $\hat{x}_{k|k}$ y $P_{k|k}$, respectivamente. Finalmente estas últimas cantidades se transformarán en la entrada del algoritmo para el tiempo $k+1$. *Imagen publicada por P. Aimonen en Wikimedia Commons, en noviembre de 2011.*

2.3.1. Filtro de Kalman Básico

El filtro de Kalman Básico [11] asume un comportamiento de sistema lineal y que las mediciones y las predicciones siguen una distribución Gaussiana.

A continuación se describen las componentes del desarrollo matemático del filtro:

- F_k : Matriz de transición de estado, de dimensiones $N \times N$.
 - H_k : Matriz de transformación de estado a medición, $K \times N$ (K corresponde al número de variables de estado medidas en un instante k , y puede ser menor a N).
 - Q_k : Matriz de covarianza del ruido del proceso ($N \times N$).
 - R_k : Matriz de covarianza del ruido de las mediciones ($K \times K$).
 - B_k : Matriz de control de entrada (contiene alteraciones que se querrían agregar al sistema de manera deliberada, por ejemplo, como la condición de parada de un vehículo en movimiento). Esta matriz es de dimensiones $N \times L$, donde L es la dimensión del vector de control de entrada u_k .

- q_k : Corresponde al ruido del proceso modelado, en las variables de estado. Posee una distribución normal multivariada y centrada en cero, con una matriz de covarianza Q_k .

Con estas variables, podemos describir las ecuaciones que explican la evolución del proceso del algoritmo de Kalman básico (un esquema de los cálculos involucrados en el proceso de estimación de estados se puede ver en la Figura 2.9):

1. Fase predictiva:

Las ecuaciones de estimación de estado y matriz de covarianza *a priori* son:

$$\hat{x}_{k|k-1} = F_k \hat{x}_{k-1|k-1} + B_k u_k, \quad (2.1)$$

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k. \quad (2.2)$$

2. Fase correctiva:

El proceso de corrección comienza con la determinación de las siguientes cantidades:

$$\hat{z}_k = H_k \hat{x}_{k|k-1}, \quad (2.3)$$

$$\tilde{z}_k = z_k - \hat{z}_k. \quad (2.4)$$

Posteriormente se calcula la matriz de covarianza entre residuos (S_k), con la que se calcula la ganancia de Kalman: K_k . Formalmente,

$$S_k = H_k P_{k|k-1} H_k^T + R_k, \quad (2.5)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1}. \quad (2.6)$$

Con la ganancia de Kalman calculada, se actualiza el valor de la estimación de estado (2.7) y la matriz de covarianza a posteriori (2.8), del siguiente modo,

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{z}_k, \quad (2.7)$$

$$P_{k|k} = (I_N - K_k H_k) P_{k|k-1}. \quad (2.8)$$

La Figura 2.9 resume el proceso de predicción (obtención de las cantidades *a priori*, $\hat{x}_{k|k-1}$ y $P_{k|k-1}$) y corrección (generación de las estimaciones *a posteriori* $\hat{x}_{k|k}$ y $P_{k|k}$) del filtro de Kalman Básico.

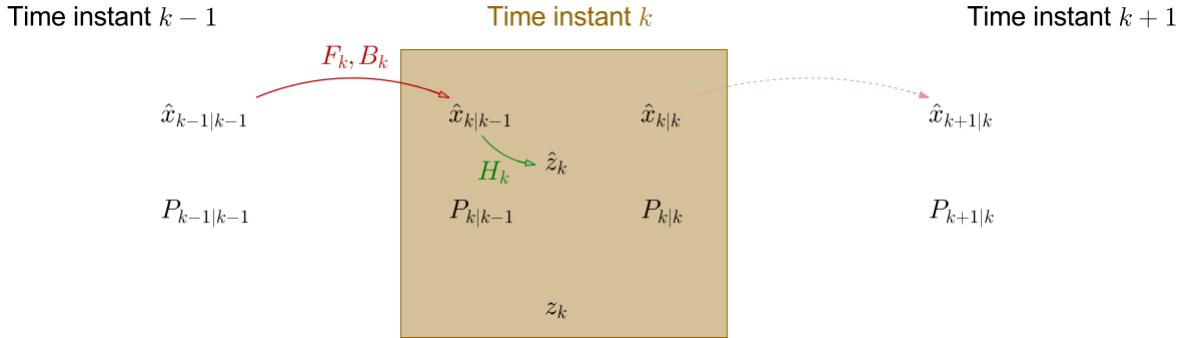


Figura 2.9: Representación del proceso de predicción (obtención de cantidades a priori) de las cantidades $\hat{x}_{k|k-1}$ y $P_{k|k-1}$; y de corrección (estimación a posteriori) para obtener las cantidades $\hat{x}_{k|k}$ y $P_{k|k}$. Para el siguiente paso, $k + 1$, estas estimaciones pasan a ser entrada (*input*) de un nuevo proceso de predicción: $\hat{x}_{k+1|k}$ y $P_{k+1|k}$. E. Matsinos, 2016. *The Kalman Filter: a didactical overview*.

Ejemplo: Partícula acelerada externamente

Consideremos un vehículo sobre un riel sin curvas y sin fricción, de tal forma que se pueda asumir un movimiento unidimensional. Inicialmente el vehículo se encontrará en una posición definida como $x_0 = 0$ y en reposo, $\dot{x}_0 = 0$, cuando súbitamente es perturbado por fuerzas externas que lo inducen a moverse. Entonces, para monitorear su movimiento se hace un muestreo cada Δt segundos. Sin embargo estas mediciones no son del todo precisas, por lo que podría desearse determinar la posición y velocidad del objeto en cada instante a través de un modelo.

En esta oportunidad, el vector de estado va a estar dado por:

$$x_k = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \quad (2.9)$$

donde x representa la posición del móvil y \dot{x} su velocidad.

Si asumimos que entre los instantes $k - 1$ y k , estas fuerzas externas causan una aceleración constante, a_k , cuya distribución es una normal de media 0 y desviación estándar σ_a , podemos aplicar las leyes físicas de movimiento, se tiene

$$x_k = F_k x_{k-1} + G_k a_k, \quad (2.10)$$

con

$$F_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix}, \quad (2.11)$$

mientras que G_k agrega la componente de aceleración sobre el sistema:

$$G_k = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \Delta t \end{bmatrix}. \quad (2.12)$$

De la Ecuación 2.10 se desprende que el término original $B_k u_k$ ha sido reemplazado por $G_k a_k$; esto se debe a que la última cantidad corresponde a perturbaciones externas, y no existen entradas de control (perturbaciones controladas o conocidas, por ejemplo, que el riel posea fricción). De esto último se deduce que $q_k = G_k a_k$.

De la expresión 2.12 es posible calcular Q_k :

$$Q_k = G_k G_k^T \sigma_a^2 = \begin{bmatrix} \frac{1}{4}\Delta t^4 & \frac{1}{2}\Delta t^3 \\ \frac{1}{2}\Delta t^3 & \Delta t^2 \end{bmatrix} \sigma_a^2. \quad (2.13)$$

En cada paso temporal Δt se realiza una medición, la cual incluye ruido, de la posición del vehículo. Sea r_k el ruido de la medición:

$$z_k = H_k x_k + r_k, \quad (2.14)$$

en donde, $H_k = \begin{bmatrix} 1 & 0 \end{bmatrix}$, y desde la cantidad r_k se calcula R_k : $R_k = E[v_k v_k^T] = [\sigma_z^2]$.

Con este sistema, podemos modelar el filtro y usar las siguientes condiciones iniciales:

$$\hat{x}_{0|0} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad (2.15)$$

$$P_{0|0} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_{\dot{x}}^2 \end{bmatrix}. \quad (2.16)$$

En donde la Ecuación 2.15 describe el estado inicial del sistema y 2.16 la matriz de covarianza inicial, la cual, dependiendo de si se conoce perfectamente la posición del objeto, todos sus valores serán cero; de no conocerse bien la posición del móvil, entonces los valores de σ_x y $\sigma_{\dot{x}}$ deberán considerarse mayores a cero.

Este modelo fue el elegido para estimar la evolución del flujo y su velocidad en la implementación de la versión básica del filtro en el programa original (y por consiguiente, en su *refactoring*).

2.3.2. Filtro de Kalman de Máxima Correntropía

El filtro de Kalman basado en máxima correntropía [5], difiere del filtro de Kalman tradicional (básico) en que no asume gaussianidad en las observaciones, considerando casos en que una señal puede ser perturbada por pulsos de ruido que sigan una distribución de cola pesada. En esta oportunidad se utiliza el *criterio de máxima correntropía* del error para el proceso de corrección.

La correntropía es una medida de similitud entre dos variables aleatorias. Supongamos, $X, Z \in \mathbb{R}$ con una distribución conjunta $F_{XZ}(x, z)$. Definimos la correntropía matemáticamente como:

$$V(X, Z) = E[\kappa(X, Z)] = \int \kappa(x, z) dF_{XZ}(x, z), \quad (2.17)$$

donde E representa al operador de esperanza y $\kappa(\cdot, \cdot)$ corresponde a un kernel Mercer invariante a desplazamientos (teorema de Mercer [9]). Para este filtro se emplea una función de kernel Gaussiana (que satisface la condición de Mercer), dado por

$$\kappa(x, z) = G_\sigma(e) = \exp\left(-\frac{e^2}{2\sigma^2}\right), \quad (2.18)$$

donde $e = x - z$ es el error.

Usualmente, en situaciones prácticas, sólo se dispone de una cantidad limitada de datos y la distribución conjunta F_{XZ} podría ser desconocida, por lo que es posible estimar la correntropía usando un estimador del promedio sobre la muestra:

$$\hat{V}(X, Z) = \frac{1}{M} \sum_{i=1}^N G_\sigma(e_i), \quad (2.19)$$

en que $e_i = x_i - z_i$, con $\{x_i, z_i\}_{i=1}^M$ siendo M el número de muestras extraídas de F_{XZ} [5].

Considerando un sistema lineal, descrito por las ecuaciones de estado (2.20) y corrección (2.21)

$$x_k = F_{k-1}x_{k-1} + q_{k-1} \quad (2.20)$$

$$\hat{z}_k = H_k x_k + r_k. \quad (2.21)$$

Donde F_{k-1} y H_k corresponden a la matriz de transición de estado y matriz de observación respectivamente; y los términos q_{k-1} y r_k corresponden a los vectores de ruido del proceso y de medición, respectivamente, de media cero y matrices de covarianza:

$$\begin{aligned} E[q_{k-1}q_{k-1}^T] &= Q_{k-1}, \\ E[r_kr_k^T] &= R_k \end{aligned} \tag{2.22}$$

Usando la media (donde $\hat{q}_k = 0$) y la matriz de covarianza prior, el proceso de predicción queda como:

$$\begin{aligned} \hat{x}_{k|k-1} &= F_{k-1}\hat{x}_{k-1|k-1}, \\ P_{k|k-1} &= F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_{k-1}. \end{aligned} \tag{2.23}$$

Luego, la corrección comienza con el cálculo de la ganancia de Kalman:

$$K_k = P_{k|k-1}H_k^T(H_kP_{k|k-1}H_k^T + R_k)^{-1}, \tag{2.24}$$

y continúa con la determinación del estado y matriz de covarianza posterior:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H_k\hat{x}_{k|k-1}), \tag{2.25}$$

$$P_{k|k} = (I - K_kH_k)P_{k|k-1}(I - K_kH_k)^T + K_kR_kK_k^T \tag{2.26}$$

donde I corresponde a la matriz identidad (Ecuación 2.26).

El modelo lineal descrito en las Ecuaciones 2.20 y 2.21 pueden reescribirse usando la media prior (2.23), obteniéndose:

$$\begin{bmatrix} \hat{x}_{k|k-1} \\ y_k \end{bmatrix} = \begin{bmatrix} I \\ H_k \end{bmatrix} x_k + \nu_k \tag{2.27}$$

$$\nu_k = \begin{bmatrix} -(x_k - \hat{x}_{k|k-1}) \\ r_k \end{bmatrix}. \tag{2.28}$$

De aquí se desprende, usando las expresiones prior (2.23), el valor esperado:

$$E[\nu_k\nu_k^T] = \begin{bmatrix} P_{k|k-1} & 0 \\ 0 & R_k \end{bmatrix} \tag{2.29}$$

$$= \begin{bmatrix} B_{k|k-1;P}B_{k|k-1;P}^T & 0 \\ 0 & B_{k|k-1;R}B_{k|k-1;R}^T \end{bmatrix} \tag{2.30}$$

$$= B_kB_k^T, \tag{2.31}$$

donde B_k puede ser obtenido a partir de la descomposición de Cholesky del término $E[\nu_k \nu_k^T]$ (las cantidades $B_{k|k-1;P}$ y $B_{k|k-1;R}$ corresponden a los valores asociados a la covarianza del estado estimado y del ruido de la medición, R_k , respectivamente).

Multiplicando la ecuación 2.27 por la izquierda por B_k^{-1} , se obtiene:

$$D_k = W_k x_k + e_k, \quad (2.32)$$

donde $D_k = B_k^{-1} \begin{bmatrix} \hat{x}_{k|k-1} \\ y_k \end{bmatrix}$, $W_k = B_k^{-1} \begin{bmatrix} I \\ H_k \end{bmatrix}$, $e_k = B_k^{-1} \nu_k$.

Proponiendo una *función de costo* basada en la función de máxima correntropía (ver Ecuación de correntropía, 2.19):

$$J_L(x_k) = \frac{1}{L} \sum_{i=1}^L G_\sigma(d_{i,k} - w_{i,k} x_k), \quad (2.33)$$

donde $d_{i,k}$ corresponde al i-ésimo elemento del vector D_k , $w_{i,k}$ es la i-ésima fila de la matriz W_k y L es la suma de las dimensiones de los vectores de estado (x_k), N y medición (y_k), N' ; es decir $L = N + N'$.

Luego, bajo el criterio de máxima correntropía, se tiene que para el estado óptimo de x_k , \hat{x}_k

$$\hat{x}_k = \arg \max_{x_k} J_L(x_k) = \arg \max_{x_k} \left(\sum_{i=1}^L G_\sigma(e_{i,k}) \right), \quad (2.34)$$

en que $e_{i,k}$ es el i-ésimo elemento de e_k :

$$e_{i,k} = d_{i,k} - w_{i,k} x_k. \quad (2.35)$$

Por tanto, la solución óptima puede obtenerse resolviendo:

$$\frac{\partial J_L(x_k)}{\partial x_k} = \sum_{i=1}^L [G_\sigma(e_{i,k}) w_{i,k}^T (d_{i,k} - w_{i,k} x_k)] = 0, \quad (2.36)$$

desde donde se tiene:

$$x_k = \left\{ \sum_{i=1}^L [G_\sigma(e_{i,k}) w_{i,k}^T w_{i,k}] \right\}^{-1} \times \left\{ \sum_{i=1}^L [G_\sigma(e_{i,k}) w_{i,k}^T d_{i,k}] \right\} \quad (2.37)$$

La Ecuación 2.37 corresponde realmente a una ecuación de *punto fijo* de x_k (por 2.35) y puede ser reescrita como:

$$x_k = f(x_k) = \left(\sum_{i=1}^L [G_\sigma(d_{i,k} - w_{i,k}x_k) w_{i,k}^T w_{i,k}] \right)^{-1} \times \left(\sum_{i=1}^L [G_\sigma(d_{i,k} - w_{i,k}x_k) w_{i,k}^T d_{i,k}] \right). \quad (2.38)$$

Un algoritmo iterativo de punto fijo puede ser obtenido usando $\hat{x}_{k,t+1} = f(\hat{x}_{k,t})$, en donde $\hat{x}_{k,t}$ denota la solución en un punto fijo en la iteración t (en el instante k).

La Ecuación 2.37 entonces puede ser expresada como:

$$x_k = (W_k^T C_k W_k)^{-1} W_k^T C_k D_k \quad (2.39)$$

en que $C_k = \begin{bmatrix} C_k^x & 0 \\ 0 & C_k^z \end{bmatrix}$, con

$$C_k^x = \text{diag}(G_\sigma(e_{1,k}), \dots, G_\sigma(e_{N,k})), \quad (2.40)$$

$$C_k^z = \text{diag}(G_\sigma(e_{N+1,k}), \dots, G_\sigma(e_{N+N',k})). \quad (2.41)$$

Con estas derivaciones es posible resumir el algoritmo de estimación por filtro de Kalman de máxima correntropía como sigue [5]:

1. Escoger un σ para el ancho del kernel apropiado, y un ϵ cuyo valor sea $0 < \epsilon < 1$ (lo suficientemente pequeño para obtener una buena convergencia). Definir un estado estimado inicial $\hat{x}_{0|0}$ y una matriz de covarianza inicial $P_{0|0}$. Definir $k = 1$.
2. Usar las Ecuaciones 2.23 para obtener $\hat{x}_{k|k-1}$ y $P_{k|k-1}$, y usar descomposición de Cholesky para obtener $B_{k|k-1;P}$.
3. Definir $t = 1$ y $\hat{x}_{k|k,0} = \hat{x}_{k|k-1}$, donde $\hat{x}_{k|k,t}$ denota el estado estimado en la iteración t del algoritmo de punto fijo.
4. Usar los siguientes pasos para calcular $\hat{x}_{k|k}$:

$$\hat{x}_{k|k,t} = \hat{x}_{k|k-1,t} + \tilde{K}_k(z_k - H_k \hat{x}_{k|k-1}), \quad (2.42)$$

con

$$\tilde{K} = \tilde{P}_{k|k-1} H_k^T (H_k \tilde{P}_{k|k-1} H_k^T + \tilde{R}_k)^{-1}, \quad (2.43)$$

$$\tilde{P}_{k|k-1} = B_{p,k|k-1} \tilde{C}_{x,k}^{-1} B_{p,k|k-1}^T, \quad (2.44)$$

$$\tilde{R}_k = B_{r,k} \tilde{C}_{y,k}^{-1} B_{r,k|k-1}^T, \quad (2.45)$$

$$\tilde{C}_k^x = \text{diag}(G_\sigma(\tilde{e}_{1,k}), \dots, G_\sigma(\tilde{e}_{N,k})), \quad (2.46)$$

$$\tilde{C}_k^z = \text{diag}(G_\sigma(\tilde{e}_{N+1,k}), \dots, G_\sigma(\tilde{e}_{N+N',k})). \quad (2.47)$$

$$\tilde{e}_{i,k} = d_{i,k} - w_{i,k} \hat{x}_{t-1,k|k}. \quad (2.48)$$

5. Posteriormente se compara la estimación de la iteración actual con la estimación de la última:

$$\frac{\| \hat{x}_{t,k|k} - \hat{x}_{t-1,k|k} \|}{\| \hat{x}_{t-1,k|k} \|} \leq \epsilon. \quad (2.49)$$

Si se cumple la relación de la Ecuación 2.49, entonces $\hat{x}_{k|k} = \hat{x}_{k|k,t}$, y se continúa con el siguiente paso; en caso contrario $t \rightarrow t + 1$, y se vuelve al paso (4).

6. Finalmente, se actualiza la matriz de covarianza posterior con

$$P_{k|k} = \left(I - \tilde{K}_k H_k \right) P_{k|k-1} \left(I - \tilde{K}_k H_k \right)^T + \tilde{K}_k R_k \tilde{K}_k^T, \quad (2.50)$$

continuando con el instante $k + 1$ en el paso (2).

2.3.3. Filtro de Kalman Unscented (UKF)

Este filtro corresponde a aquel que se pretende agregar a la familia de filtros ya desarrollada en el programa base.

1. Fase predictiva:

En esta versión del filtro [17] ya no se habla de matrices de transición de estado, F_k , ni de matrices de transformación de estado-a-medición, H_k , sino más bien de funciones diferenciables f y h respectivamente, para describir la transición de estados y la transformación de estos

a estimaciones a priori. Sin embargo, previo a estas transiciones se deben seleccionar $2N+1$ puntos representativos alrededor de $\hat{x}_{k-1|k-1}$ y evaluar estos en la función no lineal f , para obtener las estimaciones de $\hat{x}_{k|k-1}$ y $P_{k|k-1}$. Estos puntos se conocen como *puntos sigma*. En la Figura 2.10 se visualiza el proceso de predicción al momento de obtener el primer conjunto de *puntos sigma*, propagarlos usando la función f y obtener $\hat{x}_{k|k-1}$ y $P_{k|k-1}$, es decir, la matriz de estados y de covarianza *a priori*.

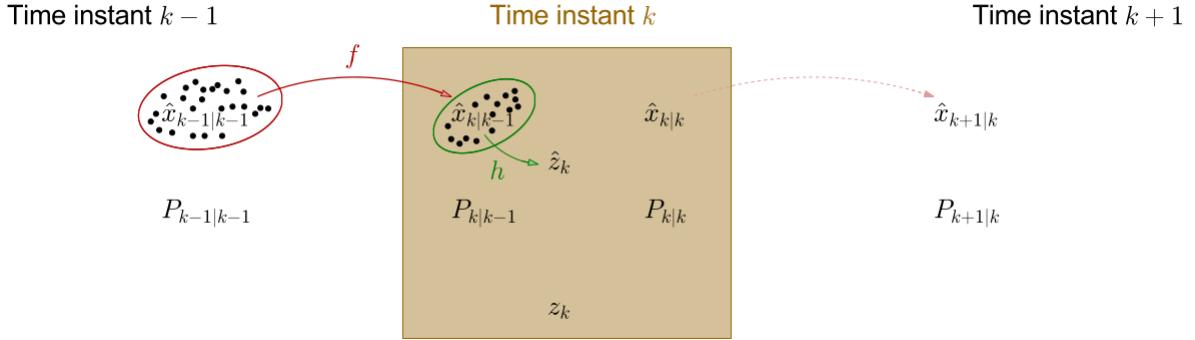


Figura 2.10: Representación del funcionamiento del filtro UKF. En esta oportunidad se hace uso de la función f y h para obtener las transformaciones $x_{k-1} \rightarrow x_k$ y $x_k \rightarrow z_k$. Esto se logra con la evaluación de los $2N+1$ *puntos sigma* generados durante la etapa de predicción (y posteriormente en la etapa de corrección). *E. Matsinos, 2016. The Kalman Filter: a didactical overview.*

La generación de los $2N+1$ puntos, se realiza a partir de la última estimación $\hat{x}_{k-1|k-1}$ de la siguiente forma:

$$\begin{aligned} \bar{x}_{k-1|k-1}^0 &= \hat{x}_{k-1|k-1} \\ \bar{x}_{k-1|k-1}^i &= \hat{x}_{k-1|k-1} + \chi_i, \quad \forall i \in [1, N] \\ \bar{x}_{k-1|k-1}^i &= \hat{x}_{k-1|k-1} - \chi_{i-N}, \quad \forall i \in [N+1, 2N], \end{aligned} \tag{2.51}$$

donde la cantidad χ_i corresponde a la i -ésima columna de la *raíz cuadrada* de la matriz:

$$(N + \lambda)P_{k-1|k-1}. \tag{2.52}$$

La matriz (2.52) puede obtenerse a partir de la descomposición de Cholesky. Por otro lado los puntos sigma se generan junto a dos conjuntos de pesos: $\{w_x^i\}$ y $\{w_p^i\}$. El primer conjunto se emplea en la estimación del estado y la predicción del estado, mientras que el segundo

conjunto es usado para obtener las matrices de covarianza. Estos pesos son definidos como:

$$\begin{aligned}
w_x^0 &= \frac{\lambda}{N + \lambda} \\
w_p^0 &= w_x^0 + 1 - \alpha^2 + \beta \\
w_x^i &= w_p^i = \frac{1}{2(N + \lambda)} \\
\sum_i^{2N} w_x^i &= 1.
\end{aligned} \tag{2.53}$$

De la Ecuación 2.53 se desprende que los pesos w_x^i son normalizados. Por otro lado, el parámetro λ se puede escribir en términos de los valores de $\alpha \in (0, 1]$ y κ según la expresión siguiente:

$$\lambda = \alpha^2(N + \kappa) - N. \tag{2.54}$$

Los parámetros α , β y κ deben ser ajustados acorde al problema que se está estudiando.

Con esto, es posible escribir las ecuaciones de la fase predictiva.

- Estimación a priori de los estados. La ecuación correspondiente es:

$$\hat{x}_{k|k-1} = \sum_{i=0}^{2N} w_x^i f(\bar{x}_{k-1|k-1}^i). \tag{2.55}$$

- Estimación a priori de la matriz de covarianza. La ecuación correspondiente es:

$$P_{k|k-1} = \sum_{i=0}^{2N} w_p^i \left(f(\bar{x}_{k-1|k-1}^i) - \hat{x}_{k|k-1} \right) \left(f(\bar{x}_{k-1|k-1}^i) - \hat{x}_{k|k-1} \right)^T + Q_k. \tag{2.56}$$

2. Fase correctiva:

Durante la fase de corrección, nuevamente se seleccionan $2N+1$ puntos representativos, alrededor de $\hat{x}_{k|k-1}$. Estos posteriormente son evaluados en la función no-lineal h .

$$\begin{aligned}
\bar{y}_{k-1|k-1}^0 &= \hat{x}_{k|k-1} \\
\bar{y}_{k-1|k-1}^i &= \hat{x}_{k|k-1} + \psi_i, \quad \forall i \in [1, N] \\
\bar{y}_{k-1|k-1}^i &= \hat{x}_{k|k-1} - \psi_{i-N}, \quad \forall i \in [N+1, 2N].
\end{aligned} \tag{2.57}$$

La cantidad ψ_i representa la i -ésima columna de la matriz de *raíz cuadrada* $(N + \lambda)P_{k|k-1}$.

Las ecuaciones del proceso de corrección, por tanto, quedan como sigue:

- Predicción de las medidas:

$$\hat{z}_k = \sum_{i=0}^{2N} w_x^i h(y_{k|k-1}^{-i}). \quad (2.58)$$

- Los residuos de las mediciones pueden obtenerse como:

$$\tilde{z}_k = z_k - \hat{z}_k. \quad (2.59)$$

- La matriz de innovación:

$$S_k = \sum_{i=0}^{2N} w_p^i (h(y_{k|k-1}^{-i}) - \hat{z}_k) (h(y_{k|k-1}^{-i})^T + R_k). \quad (2.60)$$

- La matriz de covarianza cruzada de estado a medida se describe como:

$$C_k = \sum_{i=0}^{2N} w_p^i (f(\bar{x}_{k-1|k-1}^i) - \hat{x}_{k|k-1}) (h(y_{k|k-1}^{-i}) - \hat{z}_k)^T. \quad (2.61)$$

- La ganancia óptima finalmente queda:

$$K_k = C_k S_k^{-1}. \quad (2.62)$$

- La estimación *a posteriori* de estado:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{z}. \quad (2.63)$$

- Por otro lado, la ecuación para la matriz de covarianza:

$$P_{k|k} = P_{k|k-1} - K_k S_k K_k^T. \quad (2.64)$$

Los pesos w_x^i y w_p^i son los mismos calculados en la expresión 2.53, de la fase de predicción.

Nota

En este trabajo se considerará un modelo no lineal sobre el paso del tiempo medido desde la primera observación realizada por el survey: t_0 . Es decir, la no linealidad no se aplicará sobre la estimación realizada sobre el paso anterior.

$$\begin{bmatrix} x_k(\Delta t) \\ \dot{x}_k(\Delta t) \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ 0 \end{bmatrix} + \sigma_a \begin{bmatrix} \Delta t^r \\ \frac{1}{r} \Delta t^{r-1} \end{bmatrix} = f(x_{k-1}, \Delta t) \quad (2.65)$$

Para la implementación de este filtro, se usará el modelo descrito en la ecuación 2.65. El valor de σ_a corresponde a un término similar al de una aceleración (como el modelo usado para el filtro básico), para conservar las unidades de posición y velocidad.

2.4. Laboratorio Nacional de Computación de Alto Desempeño (NLHPC)

El National Laboratory for High Performance Computing (NLHPC) es un proyecto asociativo, financiado por el PIA de CONICYT el cual dispone de un potente sistema computacional que está disponible a la comunidad científica y académica nacional (instituciones de investigación, industria y universidades), estimulando su uso en el desarrollo de áreas de investigación que requieran de herramientas computacionales robustas que deban ser usadas de manera intensiva.

El supercomputador del NLHPC disponible para la comunidad investigadora desde el 2014 en las instalaciones del Centro de Modelamiento Matemático (CMM) de la Universidad de Chile.

- 132 nodos de cómputo HP (128 nodos HP SL230 y 4 nodos HP SL250), cada uno con dos procesadores de 10 cores Intel Xeon Ivy Bridge E5-2660 V2.
- 2640 núcleos
- 6.25 TB de RAM
- 274TB de almacenamiento Lustre (DDN EXAScaler)
- 12 co-procesadores Intel Xeon Phi5110p de 2 TFlops
- Capacidad de cómputo de 70 TFlops

Para la realización de este trabajo de título se hizo uso de uno de los nodos del clúster de Leftraru, y en la ejecución de las pruebas se emplearon cuatro cores, 2400 MB por cada CPU (máxima RAM permitida) y un *job-array* de largo 93.

3

Evaluación del programa original

A continuación se ofrece una breve descripción de la rutina original y se exponen los resultados de diferentes pruebas realizadas con ella, con la finalidad de medir su desempeño computacional en términos de tiempo de ejecución y uso de memoria, a la par de estudiar los resultados en el proceso de detección de las 93 supernovas halladas por HiTS durante la campaña del año 2015.

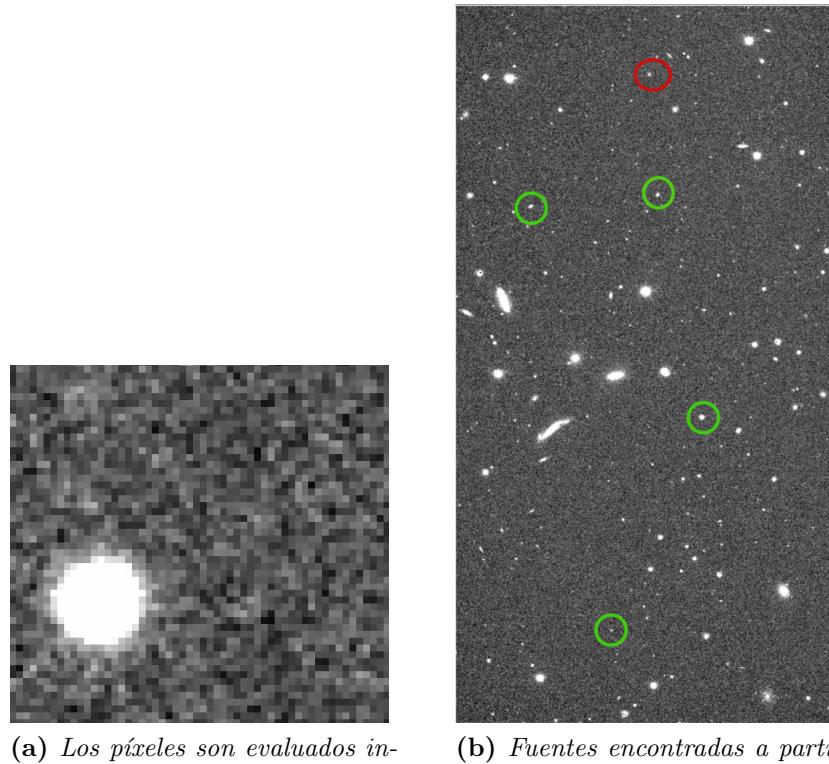
3.1. El programa

La pipeline original está estructurada por un proceso que puede dividirse en dos bloques: el primero está destinado a buscar una supernova confirmada por HiTS (cuya coordenada es conocida) y encontrar nuevos candidatos; mientras que el segundo bloque, con la lista de coordenadas de nuevos candidatos, está destinado a obtener la información¹ de estos. El proceso general se inicializa con una instancia de `RUNDATA`, la cual posee objetos de clases como `KALMANFILTER` y `SNDETECTOR` (con los cuales se configurará la rutina), y con la cual se establecerán los parámetros a emplear a través de *hard-coding*. Posterior a la configuración de parámetros se tiene una instancia de `FITSHANDLER`, para preparar las imágenes y el resto de archivos que serán usados para el proceso iterativo que se describe a continuación:

- 1. Cálculo de flujo:** A partir de las imágenes científicas y de calibración se obtiene el flujo (en `ADU`²) por píxel y es registrado en una matriz (`numpy array`) en cada iteración.
- 2. Proceso de estimación con filtro de Kalman:** En este paso se realizan los procesos de predicción y corrección para obtener una nueva estimación. Se usa alguna instancia de uno de las clases `KALMANFILTER` o `MAXIMUMCORRENTROPYKALMANFILTER` (en el programa original sólo están implementados los filtros básico y de máxima correntropía).

¹Con información se refiere a extraer muestras o estampillas desde las imágenes (de flujo, científica, etc) como también de las estructuras matriciales (matrices de estado, covarianza asociada, etc.) centradas en la coordenada del respectivo candidato.

²Analog-to-digital unit



(a) Los píxeles son evaluados individualmente.
 (b) Fuentes encontradas a partir de agrupamiento.

Figura 3.1: La figura de la izquierda, corresponde a una muestra de una imagen científica, en donde se logra observar sus píxeles los cuales deben ser evaluados individualmente primero. Luego, en la figura derecha, se destaca en un círculo rojo, la supernova conocida (SN34) y en círculos verdes las fuentes candidatas a supernova, como resultado del proceso de agrupamiento de los píxeles que satisfacen cierto conjunto de criterios.

3. Detección de fuentes: En este paso se estudia la idoneidad de los píxeles tanto del flujo como de las estimaciones de los mismos (obtenidos con el filtro de Kalman) y del resto de las imágenes al momento de verificar una serie de criterios con los cuales estos se agruparían entre vecinos para formar candidatos (este proceso se realiza en SNDETECTOR) (estos criterios son heredados en el futuro por la versión refactorizada de este programa, en la clase SOURCEFINDER). En la Figura 3.1 se ejemplifica este proceso de agrupamiento.

4. Actualización de candidatos: Revisa si hay nuevos candidatos a ser considerados. Los nuevos candidatos se registran en un arreglo (`numpy array`), mientras que los guarda la información de la supernova conocida, en caso de encontrarse.

Si en el proceso anterior se encuentran candidatos, se procede a repetir los pasos de obtención de flujos, estimaciones y filtrado de píxeles. La diferencia está en que en esta ocasión se van guardando la información de estos. El nuevo ciclo queda como sigue:

1. Repetición de los pasos anteriores 1-3

2. **Guardado de resultados:** La información de los nuevos candidatos encontrados previamente es guardada en la lista diccionarios *obj* (variable de la clase **OBSERVER**) y registrado en disco usando el formato NPZ (formato brindado por NUMPY para comprimir datos).

El diagrama de la Figura 3.2 entrega una perspectiva general de la secuencia de pasos que realiza el programa. Sin embargo, nos encontramos con otro problema de implementación: la existencia de código duplicado. Ya que ambos bloques están repitiendo los mismos pasos salvo el último en el que se diferencian en el registro de los resultados: en uno se guarda la información de la supernova conocida (sólo si la detecta) y en el segundo, sólo si hay nuevos candidatos, guarda la información de estos.

Cabe destacar que durante el proceso de estudio del programa original se encontró que la lista de archivos, que debe ser procesada en orden cronológico, de acuerdo a la época (o fecha de observación), no está siendo bien filtrada durante el proceso de selección de imágenes científicas en la clase **FITSHANDLER**, lo que puede involucrar imprecisiones en la detección de candidatos.

3.2. Estructura de datos

Debido a la naturaleza de la información de entrada (imágenes) se debe trabajar en píxeles, por lo que la estructura de datos que representen las variables de estado debe considerar la dimensión de las imágenes (teniendo en cuenta que por cada píxel se debe modelar las variables de estado flujo y velocidad de flujo, y covarianza asociada). Debido a esto, en el trabajo de Pablo Huentelemu [15] se diseñaron *hiperrectángulos* para modelar las variables de estado y covarianza de todos los píxeles de una imagen. Los datasets contienen imágenes de dimensión 4094×2046 . La Figura 3.3 muestra el esquema de sistema de matrices o *hiperrectángulos* usada para representar el estado de cada píxel y su respectiva covarianza.

3.3. Pruebas

Se realizaron dos tipos de prueba: la primera, orientada a medir tiempo de ejecución y uso de memoria principal, se llevó a cabo en un computador personal de 8 GB (DDR4) de RAM y una CPU de frecuencia 2.80 GHz sobre tres series de datos definidos por pares de CCD y campo en los que se sabe que hay una supernova. El segundo tipo de prueba se realizó en Leftraru, pasando por algunas complicaciones relacionadas con el diseño del programa ya que el código original contiene

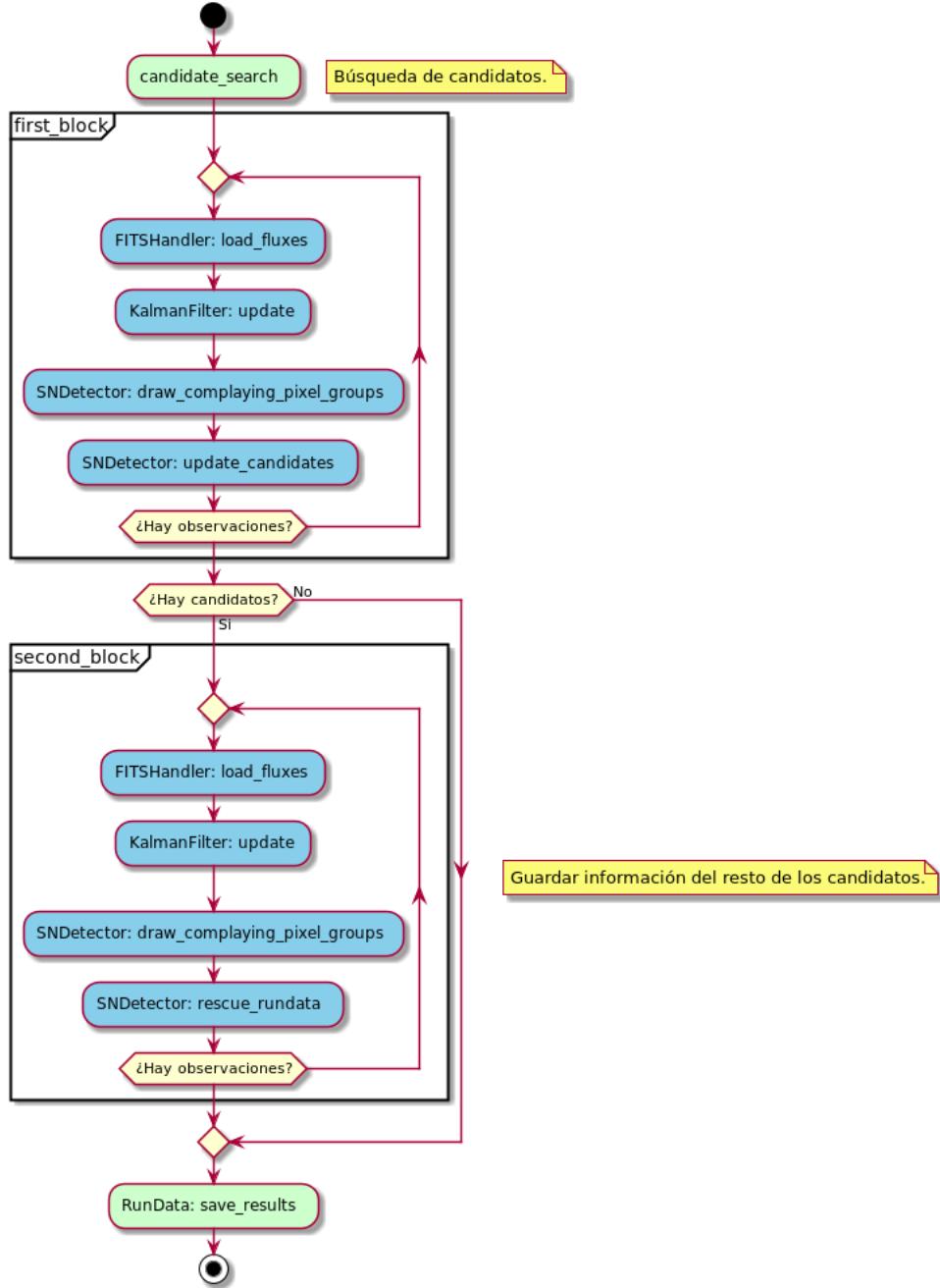


Figura 3.2: Diagrama de flujo del programa original. Se aprecian dos ciclos principales: el primero est^a destinado a la b^usqueda de una supernova en una coordenada dada y la revisi^on de nuevos candidatos; y el segundo a la revisi^on de la lista de candidatos encontrados en el primer bloque del proceso, para el guardado de la informaci^on (extracci^on de estampillas, desde cada matriz, centradas en sus coordenadas) de estos candidatos. Notar que hay pasos que se repiten en la realizaci^on de ambos an^alisis.

demasiadas lⁱneas con el comando `glob`³ de Python (y una de las razones de porqu^e se decidi^o realizar el primer tipo de prueba en un computador personal), el cual lista reiteradamente el contenido de

³<https://docs.python.org/3.5/library/glob.html>

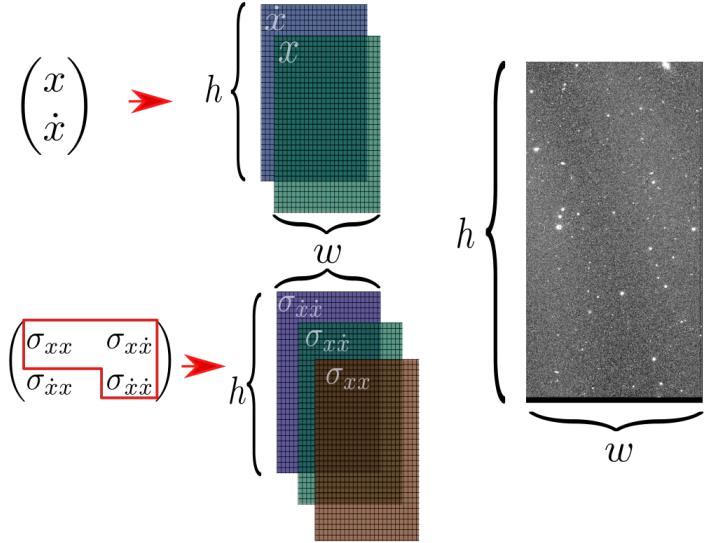


Figura 3.3: Esquema de las estructuras de datos usadas para la representación de las matrices de estados y de las matrices de covarianza de la relación entre el flujo (x) y la velocidad de flujo (\dot{x}) por cada píxel, de una imagen de ancho w y altura h . Para los conjuntos de datos, $w = 2046$ y $h = 4094$.

los directorios del *home* del usuario lo que finalmente termina saturando el nodo destinado para el lanzamiento del programa si se tiene una gran cantidad de archivos. Este problema pudo resolverse eliminando documentos resultantes de pruebas iniciales. Cabe destacar que se adaptó el programa para Python 3.6, ya que originalmente estaba para 2.7 (ambos tipos de pruebas se realizaron para la versión 3.6). En los dos tipos de prueba, se usaron los valores por defecto de los parámetros, estos son descritos en las secciones siguientes.

3.3.1. Tiempo de ejecución

El estudio del tiempo de ejecución del programa se realizó usando la función `getrusage` de la librería `resource` de Python 3.6, midiendo el tiempo de usuario en segundos. Las mediciones se realizaron sobre tres conjuntos de datos, los cuales contienen alguna supernova detectada por HiTS, y que fueron seleccionados al azar: SN14, SN18 y SN80. Cada uno de ellos comprende secuencias de 26, 23 y 18 observaciones, respectivamente.

Las Tablas 3.1 y 3.2 muestran el tiempo en segundos que toma el proceso de detección de candidatos. Se destacan como procesos separados el reconocimiento de la supernova de HiTS y posteriormente, el proceso de estudio de los nuevos candidatos, respectivamente, empleando para ambos procesos el filtro básico.

Las Tablas 3.3 y el 3.4 describen el tiempo (en segundos) tomado tanto para el proceso de

Tabla 3.1: Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación del filtro, agrupación de píxeles y filtrado de los mismos durante el período de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman básico. La última fila corresponde a la media por observación.

ID	Cálc. Flujos [s]	Aplic. KF [s]	Agrup. Píxeles [s]	Actual. Candidatos [s]
SN14	293,91	24,90	68,30	0,02
SN18	260,09	22,56	45,52	0,00
SN80	204,93	17,69	38,21	0,00
\bar{t}/Obs	11,00	0,97	2,24	0,00

Tabla 3.2: Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación de los filtros, agrupación de píxeles y filtrado de los mismos durante el período de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman básico. Se observa que para las dos últimas supernovas los tiempos son cero ya que no se encontraron más candidatos.

ID	Cálc. Flujos [s]	Aplic. KF [s]	Agrup. Píxeles [s]	Guardar resultados [s]
SN14	303,18	27,28	72,34	0,02
SN18	0,00	0,00	0,00	0,00
SN80	0,00	0,00	0,00	0,00
\bar{t}/Obs	11,66	1,05	2,78	0,00

detección de la supernova de HiTS y el posterior procesamiento de posibles nuevos candidatos (en ese orden) usando el filtro de máxima correntropía.

Tabla 3.3: Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación del filtro de Kalman, agrupación de píxeles y filtrado de los mismos durante el período de reconocimiento de la supernova correspondiente. Para esta prueba se utilizó el filtro de Kalman de máxima correntropía.

ID	Cálc. Flujos [s]	Aplic. KF [s]	Agrup. Píxeles [s]	Actual. Candidatos [s]
SN14	342,44	798,48	76,47	0,00
SN18	273,64	566,89	47,96	0,00
SN80	210,68	420,12	36,05	0,00
$\bar{t}/Obs.$	12,25	26,23	2,34	0,00

La Tabla 3.5 muestra el tiempo total comprendido por ambos subprocesos usando el filtro de Kalman básico. Se destaca la diferencia del consumo de tiempo ante la ausencia y presencia de nuevos candidatos a supernova. Por otro lado, la Tabla 3.6 muestran las mismas medidas de tiempo para el filtro de máxima correntropía. Se destaca el aumento considerable de tiempo al usar este último filtro en relación al primero.

Tabla 3.4: Resultados de tiempos de ejecución correspondientes a cálculo de flujo, estimación de los filtros, agrupación de píxeles y filtrado de los mismos durante el período de estudio de los nuevos candidatos encontrados en el paso anterior. Para esta prueba se utilizó el filtro de Kalman de máxima correntropía. Se observa que para las dos últimas supernovas los tiempos son cero ya que no se encontraron más candidatos.

ID	Cálc. Flujos [s]	Aplic. KF [s]	Agrup. Píxeles [s]	Guardar resultados [s]
SN14	307,64	680,65	65,67	0,02
SN18	0,00	0,00	0,00	0,00
SN80	0,00	0,00	0,00	0,00
$\bar{t}/Obs.$	11,83	26,18	2,53	0,00

Tabla 3.5: Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman básico. La última fila corresponde a tiempo total promedio por observación.

ID	Búsqueda SN [s]	Revisión candidatos [s]	Tiempo total [s]
SN14	387,13	402,82	789,95
SN18	328,17	0,00	328,17
SN80	260,83	0,00	260,83
$\bar{t}/Obs.$	14,55	—	—

Tabla 3.6: Tiempo de ejecución de los procesos de búsqueda de supernova de HiTS, revisión de los candidatos encontrados y tiempo total comprendido por ambos procesos usando filtro de Kalman de máxima correntropía.

ID	Búsqueda SN [s]	Revisión candidatos [s]	Tiempo total [s]
SN14	1217,39	1053,98	2068,80
SN18	888,49	0,00	888,49
SN80	666,85	0,00	666,85
$\bar{t}/Obs.$	40,83	—	—

3.3.2. Uso de memoria

La memoria ocupada por el programa se midió en términos de MiB (mebibytes) usando la librería `memory_profiler`. Posteriormente las mediciones en la unidad previamente mencionada fueron pasadas a MB (megabytes) ⁴.

La Figura 3.4 muestra el comportamiento del uso de memoria principal (en términos de mebibytes) para los tres conjuntos de datos empleados, usando el filtro de Kalman básico. Se distingue un uso más intensivo (máximo alcanzado) durante el proceso en que se usaron los datos de la SN14, la cual no sólo posee una secuencia de imágenes mayor (de 26) sino también corresponde a aquella en donde se encontraron posibles nuevos candidatos. En la Tabla 3.7 se expone el máximo consumo generado durante la ejecución de la pipeline empleando el filtro básico.

⁴1MiB \simeq 1.049MB

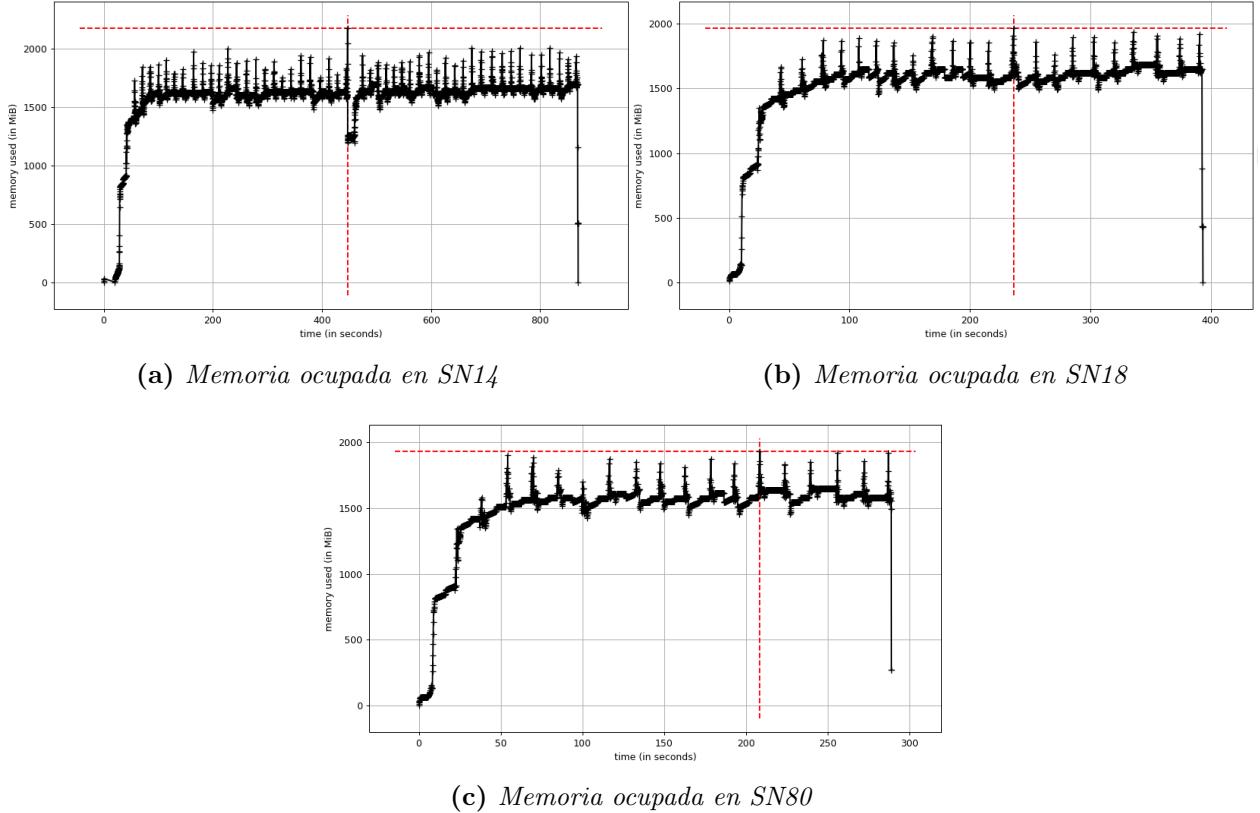


Figura 3.4: Comportamiento de la memoria (en mebibytes) durante la ejecución para los tres conjuntos de datos de las supernovas SN14, SN18 y SN80, usando el filtro de Kalman básico.

Tabla 3.7: Memoria principal (en unidades de MB) usada durante la ejecución del programa original con la versión básica del filtro de Kalman.

ID	Memoria [MB]
SN14	2282,42
SN18	2063,02
SN80	2021,27

La Figura 3.5 muestra el consumo de memoria (en mebibytes) para la ejecución del programa original para los tres conjuntos de datos, usando el filtro de máxima correntropía. Se destaca un comportamiento similar al obtenido usando el filtro básico debido a la detección de posibles candidatos con el conjunto de datos de la SN14. Por otro lado, se desprende un mayor consumo de parte de este filtro en relación a la versión básica. La Tabla 3.8 describe el consumo máximo de memoria alcanzado con el uso del filtro de máxima correntropía.

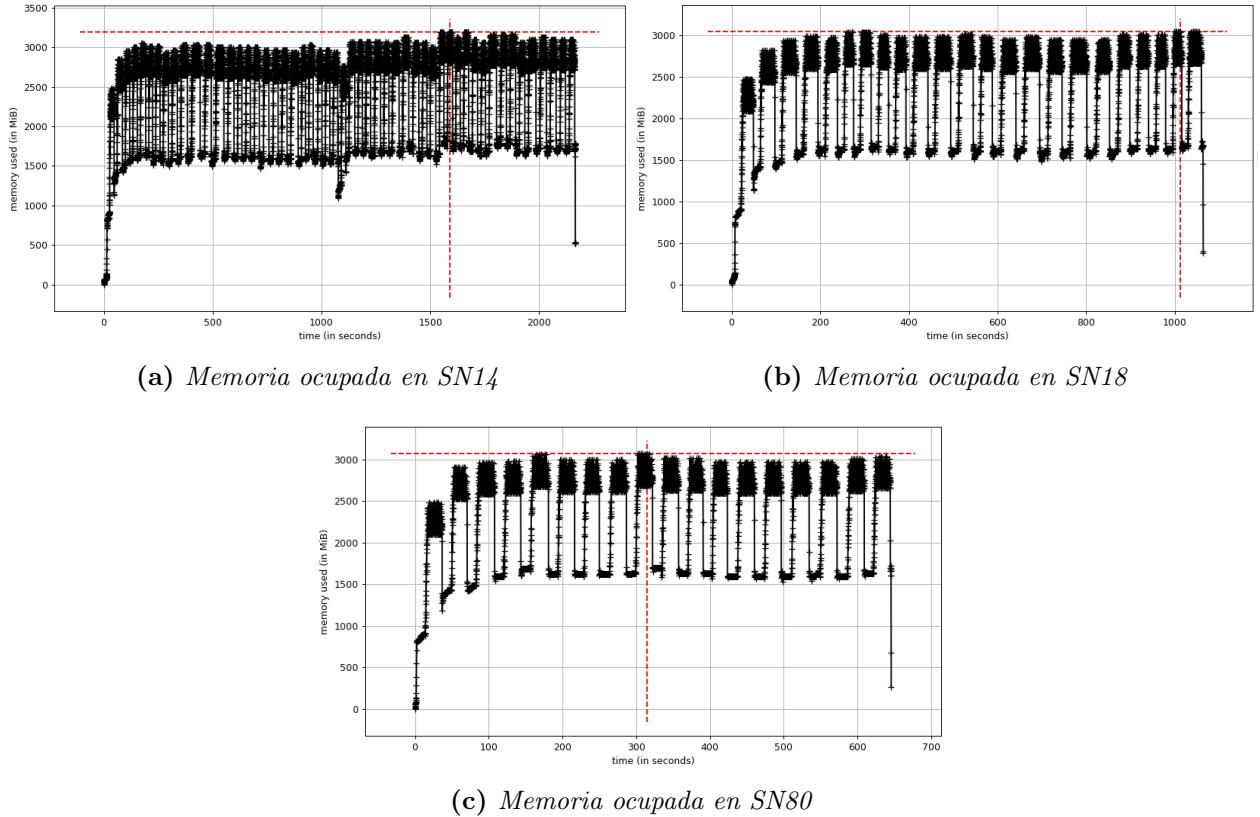


Figura 3.5: Comportamiento de la memoria (en mebibbytes) durante la ejecución para los tres conjuntos de datos de las supernovas SN14, SN18 y SN80. En los tres lanzamientos se usó el filtro de Kalman de máxima correntropía.

Tabla 3.8: Memoria principal (en unidades de MB) empleada durante la ejecución del programa original usando filtro de Kalman de máxima correntropía.

ID	Memoria [MB]
SN14	3353,13
SN18	3194,96
SN80	3209,67

3.3.3. Detección

Se realizaron las pruebas de detección sobre los conjuntos de datos de las 93 supernovas en Leftraru durante el mes de febrero de 2019, usando los umbrales de 200.0 para estimación de flujo y 50.0 para la velocidad de flujo. La Tabla 3.9, muestra el número de detecciones exitosas como verdaderos positivos (TP) así como detecciones fallidas en términos de falsos negativos (FN) y falsos positivos (FP), y escenarios en donde la ejecución del programa falló (por no encontrar archivos necesarios).

Para los parámetros usados (párrafo anterior) se obtienen resultados idénticos para ambos filtros

Tabla 3.9: Número de verdaderos positivos (TP), falsos negativos (FN) y falsos positivos (FP) (objetos no considerados por HiTS como supernova) encontrados usando cada uno de los filtros. La cuarta columna, **NaN** indica el número conjunto de datos que no se pudieron procesar por falta de algún(os) archivo(s). No se observan diferencias sustanciales entre los resultados de cada filtro. Con el filtro de máxima correntropía se encontró un falso positivo menos.

Filtro	TP	FN	FP	NaN
Básico	33	57	53	3
MCC	33	57	52	3

en términos de cantidad de supernovas detectadas. En ambos casos se detectaron 33, sin embargo existe una leve diferencia en la cantidad de falsos positivos: el programa, usando el filtro de máxima correntropía encuentra un falso positivo menos que al usar el filtro básico. Para ambos casos, tres secuencias de supernovas no pudieron ser procesadas (cuyos resultados fueron etiquetados como *NaN*) debido a la falta de imágenes de invarianza inversa (esenciales para el cálculo de flujo). Cabe destacar que para estos experimentos no se usó la optimización de Silverman, ya que su implementación estaba incompleta.

3.3.4. Observaciones

A partir de los resultados obtenidos, se procedió a estudiar las curvas de luz (cuyo flujo se mide ADU)⁵ considerando escenarios en donde supernovas conocidas son detectadas por el programa (con ambos filtros) (Figura 3.6) y donde no ocurre (Figura 3.7).

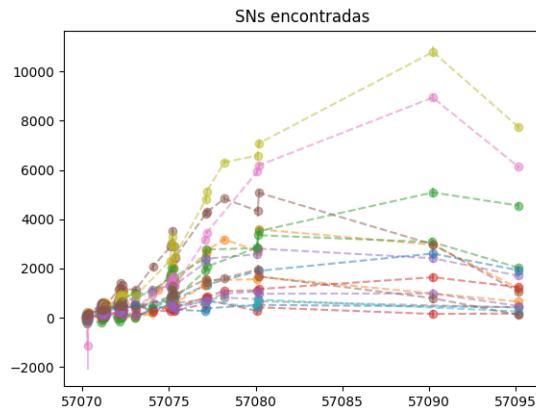


Figura 3.6: Curvas de luz (flujo en ADU vs tiempo en MJD) de 16 supernovas detectadas tanto por el filtro de Kalman básico como el de máxima correntropía. Esta figura corresponde a un análisis exploratorio de resultados obtenidos a través de métodos fotométricos tradicionales (no por filtros de Kalman), y están disponibles junto a la base de datos de HiTS [8].

⁵analog-to-digital unit

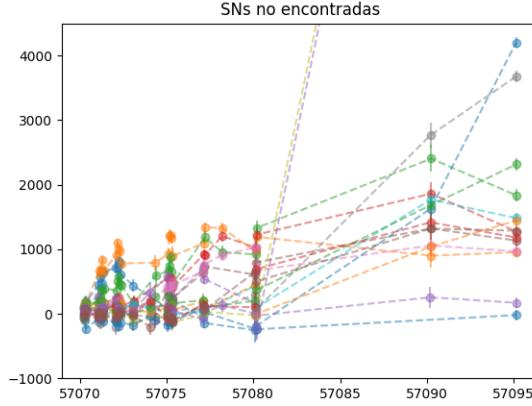


Figura 3.7: Curvas de luz (flujo en ADU vs tiempo en MJD) de 16 supernovas no detectadas ni por el filtro de Kalman básico ni por el de máxima correntropía. Esta figura corresponde a un análisis exploratorio de resultados obtenidos a través de métodos fotométricos tradicionales (no por filtros de Kalman), y están disponibles junto a la base de datos de HiTS [8]. Se aprecia el rápido crecimiento en la luminosidad de una de ellas.

En la Figura 3.6 se observa, a grandes rasgos, la presencia de una etapa de crecimiento en el flujo de los objetos (supernovas), coincidiendo con el período de alta cadencia mencionado en el capítulo anterior (Capítulo 2, subsección 2.2.1), mientras que por el contrario en la imagen 3.7 el comportamiento de los flujos es más bien irregular y no se distinguen etapas de crecimiento constante. Hay que mencionar que el algoritmo principal, aquel cuya tarea es la de filtrar y agrupar píxeles bajo criterios de descarte (implementado en SNDETECTOR), emplea un sistema de *alerta* interno con el cual se gatilla una detección cuando un grupo determinado de píxeles satisface una serie de requisitos durante cuatro épocas consecutivas. Para gatillar esta alerta, uno de los requisitos que debe cumplir un grupo de píxeles para considerarse candidato a supernova es el crecimiento continuo de los flujos por píxel. Si ocurre un descenso entonces el grupo es inmediatamente descartado.

Las Figuras 3.8 y 3.9 muestran dos casos del intervalo de tiempo en que se realizan detecciones exitosas usando ambas implementaciones de los filtros del programa original (ver Apéndice, sección 7.1 para conocer lista completa de fechas de detecciones).

Falsos positivos

Para reconocer falsos positivos se requiere contar con una herramienta externa de visualización o de *machine learning*, por ejemplo, usando un clasificador de conjuntos de píxeles que evalúe la probabilidad de que se trata efectivamente de una supernova joven. Esto se debe a que existen variados objetos como estrellas variables o *artefactos* que corresponden a zonas ruidosas de la imagen

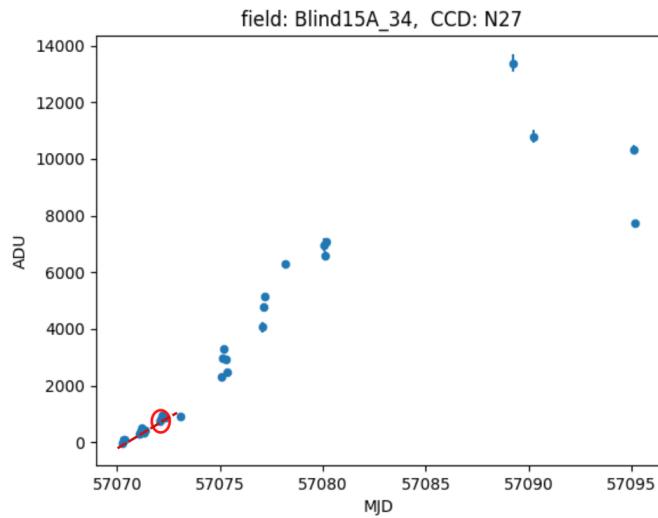


Figura 3.8: Curva de luz (en ADU vs MJD) de supernova registrada en el CCD N27, campo 34. La detección realizada por los filtros básico y de máxima correntropía se realizó en el MJD 57072.214 (época u observación 7 de 18).

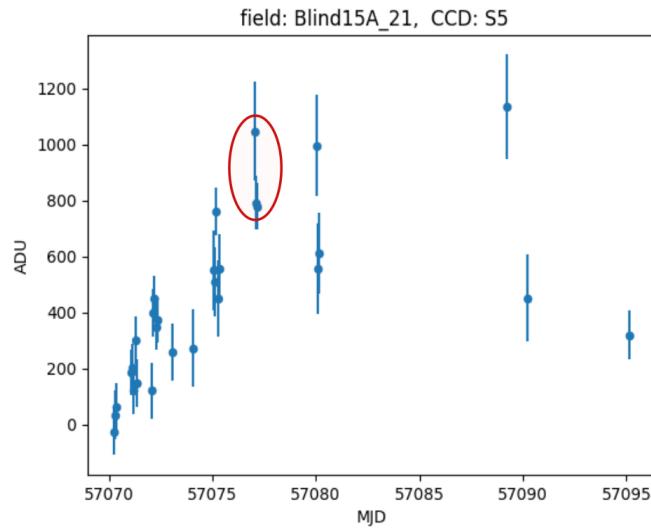


Figura 3.9: Curva de luz supernova observada en el CCD S5, campo 21. Ambos filtros detectaron la supernova en el MJD 57077.166 (época u observación 23 de 27).

científica (en particular, bordes) y pueden proveer valores alterados de flujo lo que en algunos casos puede conllevar a la detección de falsos positivos.

Una forma de poder estudiar un falso positivo es a través de la visualización de su espacio de fase y la obtención de la entropía de la curva.

4 | Refactoring

El refactoring del código original se realizó en Python 3.6. Además se mantuvieron casi la totalidad de las librerías cambiando únicamente Pymorph por Mahotas, debido a que la primera dejó de ser mantenida desde el 2010 y la segunda corresponde a la librería para Python 3.

4.1. Manejo de la rutina: RoutineHandler

La **rutina** comprende el proceso de listado de archivos (que en esta nueva versión se realiza en la clase **DATAPICKER**); los procesos iterativos de cálculo de flujo, obtención de estimaciones y la detección de candidatos con **SOURCEFINDER**; y el guardado de la información de los candidatos y/o la generación de visualizaciones (dependiendo del tipo de ejecución). Estos procesos son administrados por la clase **ROUTINEHANDLER** el cual debe instanciarse recibiendo tres archivos descritos a continuación:

- Lista de campos, CCDs, semestres y, alternativamente, las coordenadas de algún objeto de interés, en un archivo CSV (en este mismo orden), teniendo como encabezado **Field**, **CCD**, **Semester**, **POS_Y**, **POS_X**. En caso de no adjuntar las coordenadas en los campos de posición, se puede llenar con un guión ('-'). En el constructor este archivo se recibe como **obs_index_path** (ejemplo en Apéndice [D](#)).
- Diccionario de directorios y expresiones regulares de las ubicaciones de los archivos y sus nombres, respectivamente (archivo de formato CSV). En el constructor de la clase este archivo se denomina como **routes_templates** (ejemplo en Apéndice [B .1](#)).
- Diccionario de umbrales y parámetros relevantes en la ejecución del programa, así como el tipo de filtro a usar (archivo de formato CSV). En el constructor se denominó **settings_file** (ver ejemplo en Apéndice [C .1](#)).

Además de estos archivos se debe especificar el índice de la fila a procesar de la lista de campos, CCDs y semestre (primer archivo de entrada). Esto se hizo así con la finalidad de facilitar

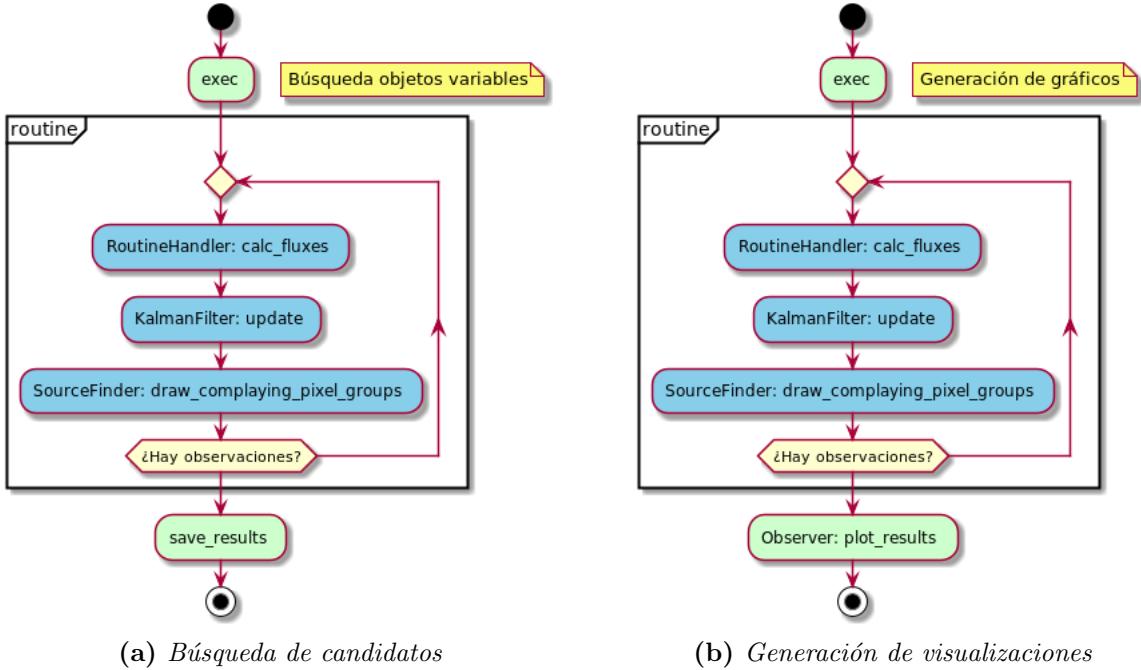


Figura 4.1: Rutina del programa refactorizado. Cuenta con dos modos de ejecución: uno de búsqueda de candidatos (izquierda) y otro de generación de gráficos (derecha). Pueden ejecutarse ambas formas comenzando con la búsqueda de candidatos y continuar con el modo de generación de gráficos. El usuario tiene la libertad de escoger el modo de ejecución.

la paralelización de los análisis de diferentes conjuntos de datos. Por lo tanto el constructor de ROUTINEHANDLER queda como sigue:

```
RoutineHandler(obs_index_path, routes_templates, settings_file, index)
```

El método más importante de esta clase es `routine`, el cual implementa la pipeline principal del programa original. Este método puede ser ejecutado de dos formas, dependiendo de la finalidad que busca el usuario: si se desea realizar una búsqueda de candidatos (considerando también las coordenadas entregadas en `obs_index_path`) o, si se desea visualizar información de candidatos ya encontrados (a partir de una lista de coordenadas cargadas desde un archivo NPZ). La Figura 4.1 muestra ambas formas de ejecución. La lista completa de métodos de esta clase se encuentran en el Apéndice, sección E .

4.2. Manejo de datos de entrada

Todas las imágenes de entrada son manipuladas y servidas por la clase DATAPICKER. Esta clase se inicializa recibiendo un *path* hacia el archivo de configuración `routes_templates` que contiene tanto las rutas a los archivos así como los nombres de estos en términos de expresiones regulares;

semestre a los que corresponde la secuencia de observaciones (los dos últimos dígitos del año concatenados con la letra A en caso de corresponder al primer semestre o B al segundo); el campo (representado como un número de dos dígitos, comenzando con cero para valores menores a 10) y el detector CCD (cadena de tres caracteres donde el primero de ellos describe a que grupo de detectores corresponde: N o S (ver Figura 2.5); además de un número entero que va de 1 a 36) como strings (para más detalles, ver Apéndice, sección B).

Es en DATAPICKER donde se realiza el proceso de filtrado de las imágenes (descarte por airmass) y su posterior ordenamiento, esta vez considerando un filtrado más minucioso, considerando la correspondencia de las fechas MJD contenidas en los header de cada archivo FITS. Para ver los detalles de este proceso de listado de imágenes y filtrado, revisar el Apéndice, sección F .

4.3. Determinación de flujos

El cálculo del flujo, en este refactoring, se independizó del manejo de archivos (en el programa original estaba alojado en la clase FITSHANDLER) y se implementó en el script `utils` pensado como librería (una descripción detallada de los métodos se encuentra en el Apéndice, sección H).

4.4. Filtros originales

La refactorización de los filtros de Kalman originales implicó la implementación de nuevas clases e interfaces para el desarrollo del patrón propuesto: Strategy (ver familia de métodos resultante en la Figura 4.2). A continuación se presentan cada una de ellas:

- **IPredict:** Interface que describe el comportamiento de la función PREDICT de un filtro de Kalman. El método `predict` recibe como parámetros el paso de tiempo (Δt), la matriz de estado, la matriz de covarianza de estado, y las predicciones de las matrices de estado y covarianza determinadas en el paso anterior (con la finalidad de actualizar estas variables). Su firma queda como:

```
predict(delta_t, state, state_cov, pred_state, pred_cov)
```

Este método entrega finalmente las matrices de estado y covarianza de estado predicho.

- **ICorrect:** Interface que describe el comportamiento de la función CORRECT de un filtro de Kalman. El método `correct` recibe como parámetros la matriz de flujo (`z`, correspondiente

a la observación) y de varianza de la observación (R), las matrices de estado y covarianza predichas y las matrices de estado y covarianza estimadas obtenidas en el paso anterior y que serán actualizadas. Su firma queda como:

```
correct(z, R, pred_state, pred_cov, state, state_cov)
```

Esta función entrega finalmente las matrices de estado y covarianza de estado corregido.

4.4.1. Predicción

LinearPredict: Clase que extiende IPredict. Implementa el método `predict` que será usado tanto por el filtro básico como por el de máxima correntropía. Su instanciación recibe como argumento `sigma_a` (desviación estándar del ruido del proceso, asumiendo una distribución gaussiana para las observaciones).

4.4.2. Corrección

BasicCorrect: Clase que extiende ICorrect. Implementa el método `correct` que será usado para el tipo de filtro de Kalman básico.

MCCorrect: Clase que extiende ICorrect. Implementa el método `correct` que será usado para el tipo de filtro de Kalman de máxima correntropía. El constructor de esta clase recibe los siguientes parámetros:

- **epsilon:** Cantidad con la cual se contrastará el error o precisión que se quiera lograr con la estimación (el valor definido por defecto en el programa original es de 10^{-6}).
- **max_iter:** Número máximo de iteraciones (de acuerdo al código original, se estableció como 10 su valor por defecto).
- **silverman:** *boolean*. Determina si se usa o no la aproximación de Silverman para el ancho de banda del kernel Gaussiano (por defecto es *false*).
- **std_factor:** Factor de desviación estándar usado en la aproximación de Silverman (su valor predeterminado es de 100, 0).
- **sigma:** Sigma usado para el ancho de banda en el cálculo de la correntropía. Puede ser o no optimizado por Silverman. Por defecto, es 1000, 0, según programa original.

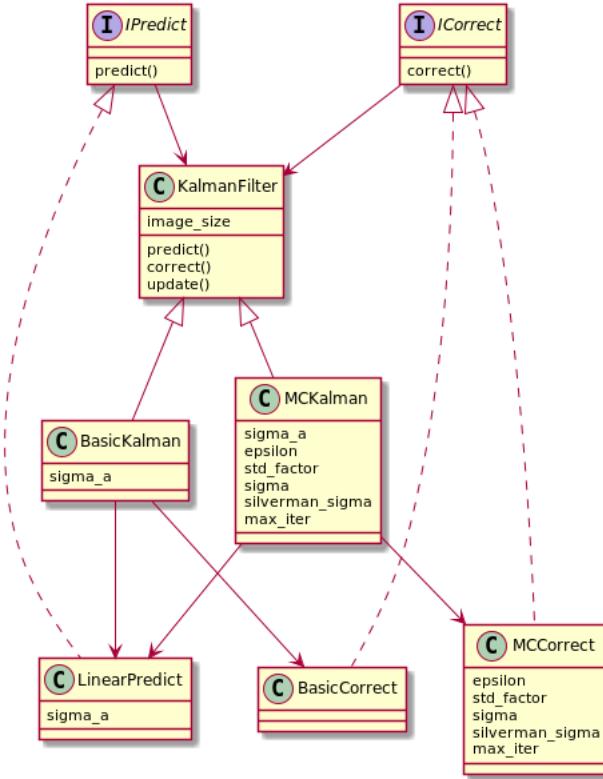


Figura 4.2: Familia de filtros de Kalman y patrón *strategy* usado en la implementación de los métodos `predict` y `correct`.

4.4.3. Filtros refactorizados

- **KalmanFilter:** Clase abstracta padre de los subtipos `BasicKalmanFilter` y `MCKalmanFilter`. Posee los métodos abstractos `predict` y `correct`, que son definidos de acuerdo a las estrategias de predicción y corrección descritas previamente.
- **BasicKalmanFilter:** Representa el filtro básico de Kalman. Está compuesto por las estrategias `LinearPredict` y `BasicCorrect`.
- **MCKalmanFilter:** Representa el filtro de máxima correntropía. Está compuesto por las estrategias `LinearPredict` y `MCCorrect`.

4.5. Detección de candidatos

Mientras que la detección de candidatos en el programa original se realiza en una instancia de la clase `SNDETECTOR`, la detección en la nueva versión se realiza en `SOURCEFINDER`, el cual funciona de la misma forma que su antecesor. El cambio de nombre representa la intencionalidad de extender

la funcionalidad de esta clase no sólo a la detección de supernovas sino además encontrar objetos como estrellas variables.

Su constructor requiere los siguientes argumentos:

- **flux_thresh**: Umbral de corte para el flujo estimado por el filtro de Kalman.
- **flux_rate_thresh**: Umbral de corte de velocidad de flujo estimado por el filtro de Kalman.
- **rate_satu**: Tasa de saturación en la velocidad de flujo.
- **accum_neg_flux_depth**: Cantidad de épocas de registro de píxeles negativos (para la construcción de una matriz de *booleans* de esta profundidad).
- **accum_med_flux_depth**: Cantidad de épocas de registro de píxeles cuya intensidad mediana (durante las épocas) es mayor a 1500.
- **image_size**: Dimensión de las imágenes FITS (por defecto, corresponde a la tupla (4094, 2046)).
- **n_consecutive_obs**: Número de alertas u observaciones consecutivas a considerar para confirmar una detección. Para las versiones antigua y nueva de este programa, se estableció como 4.

Los métodos se encuentran descritos en el Apéndice, sección [G](#). Además, cabe mencionar que los parámetros previamente mencionados se ajustan al archivo de configuración, `settings_file`.

4.6. Visualización de resultados

En el proceso de visualización participan dos clases: `OBSERVER` y `VISUALIZER`. La primera clase es la encargada de generar una lista de diccionarios dentro de los cuales se almacena información de los candidatos encontrados durante el proceso de detección. Esta lista es guardada como una variable de instancia en `OBSERVER` denominada `objects` y la información contenida por cada uno de los diccionarios corresponde a las siguientes componentes:

- Ubicación del objeto en la primera imagen científica procesada, como un arreglo de *floats* de largo dos.
- Lista de épocas en las que el objeto fue detectado.

- Listas de estampillas de diferente profundidad, de ancho y alto 21×21 . Las estampillas de cada lista tendrán una profundidad propia dependiendo desde la estructura que se extraigan. Estas estructuras pueden ser imágenes como la científica, de diferencia, máscara, etc., o a estructuras como las etiquetas de píxeles grupales e individuales, matrices de estado, etc. Cada una de las estampillas de cada lista corresponderá a la medición, cálculo o estimación obtenida para una época específica. Además cada una de las estampillas está centrada en la coordenada del objeto candidato a revisar.

Para el registro de estos datos se emplean dos métodos de `OBSERVER` listados a continuación:

- `set_space (cand_data)`:

Recibe lista de candidatos (coordenadas) `cand_data` y crea lista de diccionarios, `objects`, con la finalidad de guardar la información de los primeros para todas las épocas analizadas creando arreglos (de la librería NUMPY) de estampillas 21×21 destinadas a guardar los datos de las diferentes estructuras: imágenes, matrices de estado estimado y predicho, covarianzas por pixel, matrices de etiquetas de píxeles, etc. Cada estampilla estará centrada en la posición del candidato en la primera imagen científica analizada.

- `look_candData(cand_data, pred_state, pred_state_cov, kalman_gain, state, state_cov, time_mjd, flux, var_flux, science, diff, psf, base_mask, dil_base_mask, pixel_flags, pixel_group_flags, mjd_idx)`:

Con el espacio generado en la estructura `objects`, se procede a ejecutar la pipeline principal (cálculo de flujo, generación de estimaciones con filtro de Kalman y proceso de detección) para, en esta ocasión, ir guardando resultados de los candidatos en `cand_data` en su diccionario respectivo por cada época (cuyo índice está representado por el argumento `mjd_idx`). Por tanto recibe como argumentos las matrices de estados y covarianzas predichos (`pred_state` y `pred_state_cov` respectivamente), las matrices de estados y covarianzas estimados (`state` y `state_cov`) la matriz de ganancia de Kalman (`kalman_gain`), matriz de flujo calculado y varianza asociada (`flux` y `var_flux`), imagen científica (`science`) y de diferencia (`diff`), imagen de la PSF usada (`psf`), matrices de etiquetas de píxeles (`pixel_flags` y `pixel_group_flags`) e imagen de la máscara usada (`base_mask`) junto a su versión post-proceso de dilatación (`dil_base_mask`).

- `plot_results(semester, field, ccd, plots_path)`

Finalmente, con la variable de instancia `objects` generada, es posible crear los gráficos de

cada candidato con este método, el cual recibe como parámetros el semestre (`semester`), el campo (`field`) y el CCD (`ccd`) de donde se obtuvieron las imágenes junto a la ruta al directorio donde se quiere guardar las imágenes (`plots_path`) como argumentos. Los gráficos son generados por una instancia de la clase `VISUALIZER`.

La clase `VISUALIZER` permite la obtención de tres tipos de gráficas de acuerdo al método llamado.

- **Curvas de luz, estados, etiquetas y varianzas**

Esta gráfica contiene cuatro series de tiempo cuyos datos se obtienen a partir del pixel ubicado justo en el centro del objeto considerado candidato. La primera serie de tiempo corresponde al flujo estimado y predicho por Kalman junto al flujo observado o medido. La segunda serie de tiempo corresponde a la componente de velocidad de flujo estimada y predicha de los estados obtenidos por Kalman. La tercera serie de tiempo corresponde a la evolución de las etiquetas (criterios que no satisface el pixel). Finalmente, la cuarta serie de tiempo, muestra la evolución de las varianzas de cada elemento de estado estimado y predicho, además del correspondiente a la medición.

- **Estampillas**

Muestra el comportamiento de los píxeles en las estampillas de dimensión 21×21 de las siguientes estructuras: imagen científica, PSF, flujos observado y su varianza, estados de flujo y su velocidad estimados por filtro de Kalman, etiquetas grupales e individuales de píxeles.

- **Curva de estado**

Esta curva se logra a partir de los valores estimados de flujo y de la velocidad de esta obtenidos por el filtro de Kalman. Esta gráfica muestra la complejidad de la curva visualizada calculando su entropía [3].

Estas gráficas son generadas gracias a los siguientes métodos de `VISUALIZER`:

- `print_lightcurve(obj, obs_rad, height, width, save_filename):`

Este método está destinado a crear series de tiempo, para contrastar, de diferentes variables de interés tales como el flujo observado, estimado y predicho (medidos en el pixel central del candidato) visualizados en la misma gráfica. Del mismo modo, en un gráfico dispuesto bajo el primero, se dibujan las series de tiempo de las velocidades estimadas y predichas. Posteriormente se grafica la evolución de las etiquetas de píxeles individuales y grupales, y el último gráfico generado corresponde a la serie de tiempo de las varianzas y covarianzas de las

diferentes variables como flujo, estimaciones de flujo y sus predicciones obtenidas con el filtro de Kalman, etc.

Los parámetros `height` y `width` corresponden a la altura y ancho de la imagen generada. Por último, `save_filename` corresponde al nombre con el que se guardará el archivo en disco en formato PNG. Ejemplo de imagen resultante en Figura 4.3.

- `print_stamps(obj, height, width, save_filename):`

Recibe como entrada un diccionario de la lista de objetos almacenados por una instancia de `OBSERVER`. Con esta función las estampillas son impresas secuencialmente (orden cronológico) en filas, donde cada una de estas últimas corresponde a algún tipo de imagen o estructura como la imagen científica, estimación de flujo, etiquetas de píxeles, etc. (ver ejemplo de imagen en la Figura 4.4).

Los valores `height` y `width` corresponden a la dimensión de la imagen resultante. El argumento `save_filename` indica el nombre con el que se quiere guardar el documento (de formato PNG) en disco.

- `print_space_states(obj, obs_rad, height, width, flux_thresh, rate_flux_thresh, save_filename):`

Esta función es la encargada de graficar la curva de estados por los que pasa un candidato (cuya información está contenida en el diccionario `objects`). Se visualizan los estados en su detección y tres épocas anteriores previas a su confirmación (durante las alertas). Estos estados están definidos por el flujo y la velocidad de flujo estimados por el filtro de Kalman. Además en la misma imagen, se indica el nivel de complejidad de la curva en términos de entropía [3].

El cálculo de la complejidad de la curva (o entropía de la curva) se obtiene a partir de la siguiente expresión:

$$H(\Gamma) = \log \left(\frac{2L}{C} \right), \quad (4.1)$$

donde L corresponde al largo de la curva Γ y C es el largo de su envolvente convexa (*convex hull*) [3]. Un ejemplo de imagen resultante se observa en la Figura 4.5, en donde el valor de la entropía de la curva se muestra en la leyenda.

Los argumentos del método, `height` y `width`, corresponden a las dimensiones de alto y ancho de la imagen a generar, mientras que `save_filename` indica el nombre del archivo generado a guardar.

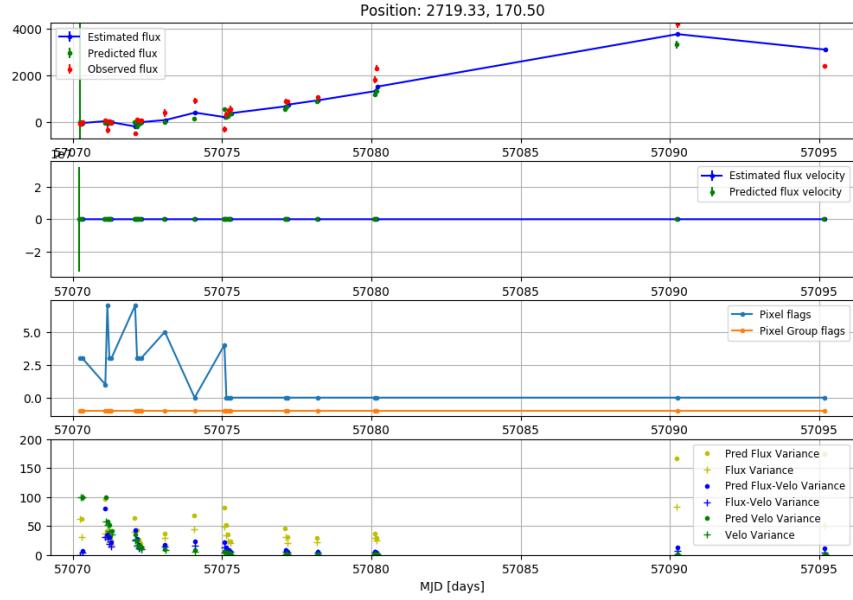


Figura 4.3: Conjunto de series de tiempo de diferentes componentes de interés en el pixel ubicado en la coordenada de posición del candidato. El primer gráfico (de arriba hacia abajo) muestra la evolución del flujo medido en contraste con el comportamiento de la predicción y estimación realizada por el filtro de Kalman del mismo flujo durante las épocas de las observaciones. La segunda gráfica muestra los cambios de la predicción y estimación de la velocidad de flujo (obtenidas por el filtro de Kalman) en el tiempo. El tercer esquema muestra la evolución de las etiquetas (grupal e individual) del pixel ubicado en la coordenada del candidato. Finalmente, el último gráfico visualiza el comportamiento de las diferentes varianzas y covarianzas tanto de las componentes predichas y estimadas por el filtro (flujo y velocidad de flujo), así como de las mediciones del mismo flujo (observado).

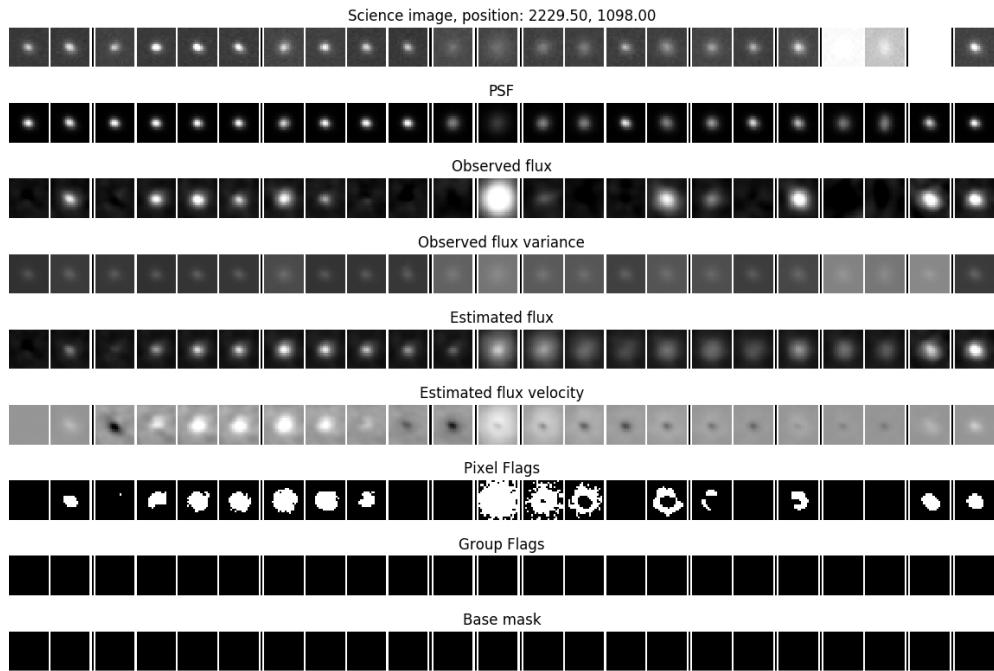


Figura 4.4: Estampillas de matrices de 21×21 píxeles y etiquetas que describen su comportamiento a través del tiempo: la primera fila de imágenes corresponde a estampillas obtenidas desde las imágenes científicas en donde debiese habitar la supernova observada durante todas las observaciones. La segunda fila muestra los diferentes modelos de PSF obtenidos para diferentes épocas. La tercera fila muestra el flujo observado en la misma posición. Le sigue la varianza de este flujo. Posteriormente viene el flujo estimado por el filtro de Kalman siguiendo la velocidad de flujo estimado. Luego vienen las etiquetas de los píxeles reconocidos por el programa como pertenecientes a un objeto transitorio (etiquetado por pixel y por grupo de píxeles). La última fila corresponde a la máscara base usada durante el análisis.

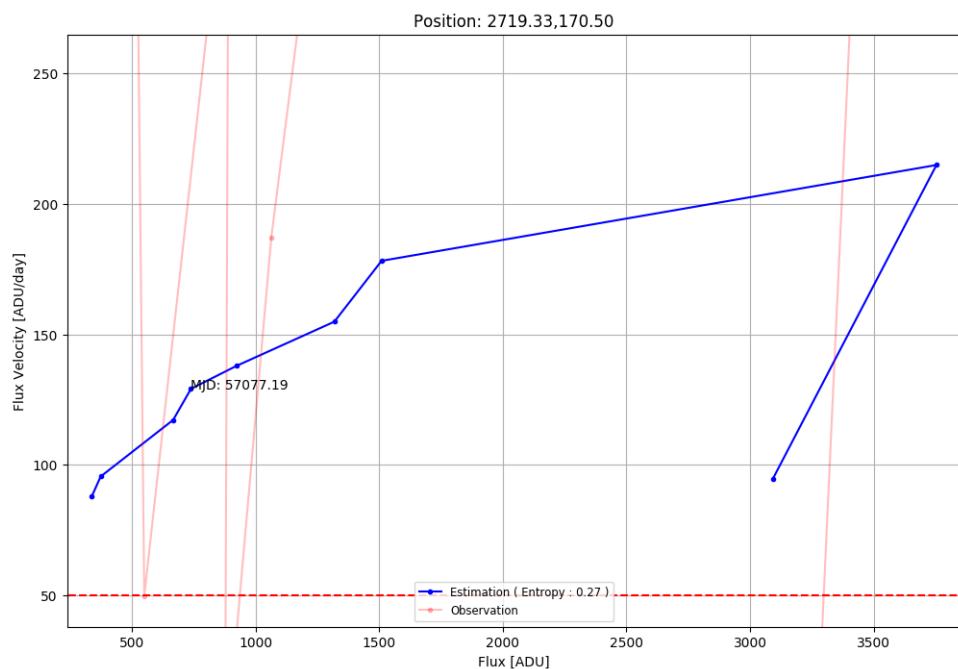


Figura 4.5: Espacio de fase de flujo y velocidad de flujo de un candidato. En azul se destaca la estimación lograda por filtro de Kalman y en rojo el flujo observado versus la misma estimación de velocidad. Notar que en la leyenda de la figura, se indica el nivel de complejidad de la curva estimada en términos de su entropía [3] para la curva de flujo y velocidad de flujo estimado.

5

Filtro de aproximación no lineal: unscented Kalman

En este capítulo se describe la estructura y detalles importantes del nuevo miembro de la familia de clases de filtros de Kalman: UNSCENTEDKALMAN

5.1. Diseño

El diseño de la nueva clase de filtro se realizó continuando con la arquitectura del patrón Strategy, definiendo los métodos de predicción y corrección en instancias de clases que implementen las interfaces IPREDICT e ICORRECT respectivamente: UNSCENTPREDICT y UNSCENTCORRECT.

El filtro unscented se desarrolló en la clase UNSCENTEDKALMAN, la cual encierra los métodos de predicción y corrección (descritos en el Capítulo 2, subsección 2.3.3) implementados en `predict` y `correct`.

El constructor de esta clase requiere como argumentos de entrada los siguientes parámetros:

- `f_func`: Función usada para obtener el primer conjunto de puntos sigma para el período de predicción. Consiste en una función vectorial que se aplica sobre cada elemento de una matriz cuya dimensión es la misma que la de la matriz de estado. Corresponde a la función principal con la que se aplicará el modelo no lineal en el proceso de predicción.
- `h_func`: Función usada para obtener el segundo conjunto de puntos sigma durante el proceso de corrección. Su aplicación procede de la misma forma que la descrita para `f_func`. Participa en el proceso de corrección.
- `f_args`: Corresponde a una lista de números reales de largo dos. El primer valor corresponde a la potencia o grado de la función mientras que el segundo término corresponde al factor que la multiplica. Estos valores están asociados a la función `f`.

- **h_args**: Corresponde a una lista de número reales de largo dos. El primer valor corresponde a la potencia o grado de la función mientras que el segundo término corresponde al factor que la multiplica. Estos valores están asociados a la función **h**.
- **alpha**: Término que indica que tan separados se encuentran los puntos sigma en torno a la media (junto a **kappa**). Por lo general su valor oscila entre 10^{-4} y 1.
- **beta**: Describe el valor de β el cual incorpora conocimiento *a priori* sobre la distribución de las variables de estado. Para distribuciones gaussianas, tiene un valor de $\beta = 2$.
- **kappa**: Corresponde a un parámetro de escalamiento secundario en la distancia de los puntos sigma a la media. Su valor va entre 0 y $3 - N$ [18].
- **sigma_a**: Varianza asociada a la variable de control Δt . La función no lineal se aplicará sobre el paso del tiempo.
- **image_size**: Dimensiones de la imagen de flujo. Corresponde a una tupla de dos números enteros (por defecto, (4094, 2046)).

Al inicializar una instancia de esta clase, se calculan tanto los pesos como el término λ , los cuales dependen de los parámetros: β , κ α y N (o cantidad de variables de estado). Ver Ecuaciones 2.53 y 2.54.

Como parte de la familia de KALMANFILTER posee el método **update**, desde el cual se llama a **predict** y a **correct**, conservando la firma.

5.1.1. Predicción

La predicción de este filtro se implementó, como se mencionó anteriormente, en la clase UNSCENTEDPREDICT (que implementa a IPREDICT). Para instanciar esta clase se debe entregar como argumento al constructor: un puntero a función (que puede o no ser lineal) **f_func** y sus argumentos en **f_args**; los pesos asociados a la media, **W_m**, y a la covarianza, **W_c**; el coeficiente λ (**lambda_**); la cantidad de variables de estado medidas, N (en este modelo, se trata de dos: flujo y velocidad de flujo, por ende $N = 2$); la varianza σ_a (**sigma_a**) y la dimensión de las imágenes **image_size**. Por lo tanto la firma del constructor queda como sigue:

```
UnscentedPredict(f_func, f_args, Wm, Wc, lambda_, N, sigma_a, image_size)
```

Se destaca que tanto los pesos como el valor de λ se mantienen constantes durante todo el proceso de predicción y estimación, valores con los cuales se obtienen los puntos sigma y el valor medio al propagar la función `f_func` (ver Ecuación 2.51).

La salida de este método está compuesta por la matriz de estado predicha, la predicción de la covarianza y los puntos sigma guardados en la variable `Xs` de la implementación.

5.1.2. Corrección

El proceso de corrección está implementado en la clase `UNSCENTEDCORRECT` la cual implementa la interfaz `ICORRECT` redefiniendo el método `correct` para esta versión del filtro. De acuerdo al proceso de corrección para el funcionamiento del filtro se deben definir las funciones $f(\cdot)$ y $h(\cdot)$, además requiere de los pesos calculados previamente (Ecuación 2.53).

El constructor de esta clase posee la siguiente firma.

```
UnscentedCorrect(f_func, h_func, f_args, h_args, Wm, Wc, lambda_, N, image_size)
```

Durante la corrección se escogen nuevamente $2N + 1$ puntos sigma en torno al estado predicho (fase previa) (ver Ecuación 2.57), sin embargo los valores de los pesos y de λ son los mismos usados durante la fase de predicción, cambiando sólo la función que se propaga sobre este nuevo conjunto de puntos: $h(\cdot)$ (expresada como `h_func`). Las operaciones que participan en este proceso se implementaron en el script `unscented_utils` (para ver detalles de los métodos, ver Apéndice, sección I).

La familia de filtros resultantes de la agregación de este nuevo módulo se puede observar en la Figura 5.1.

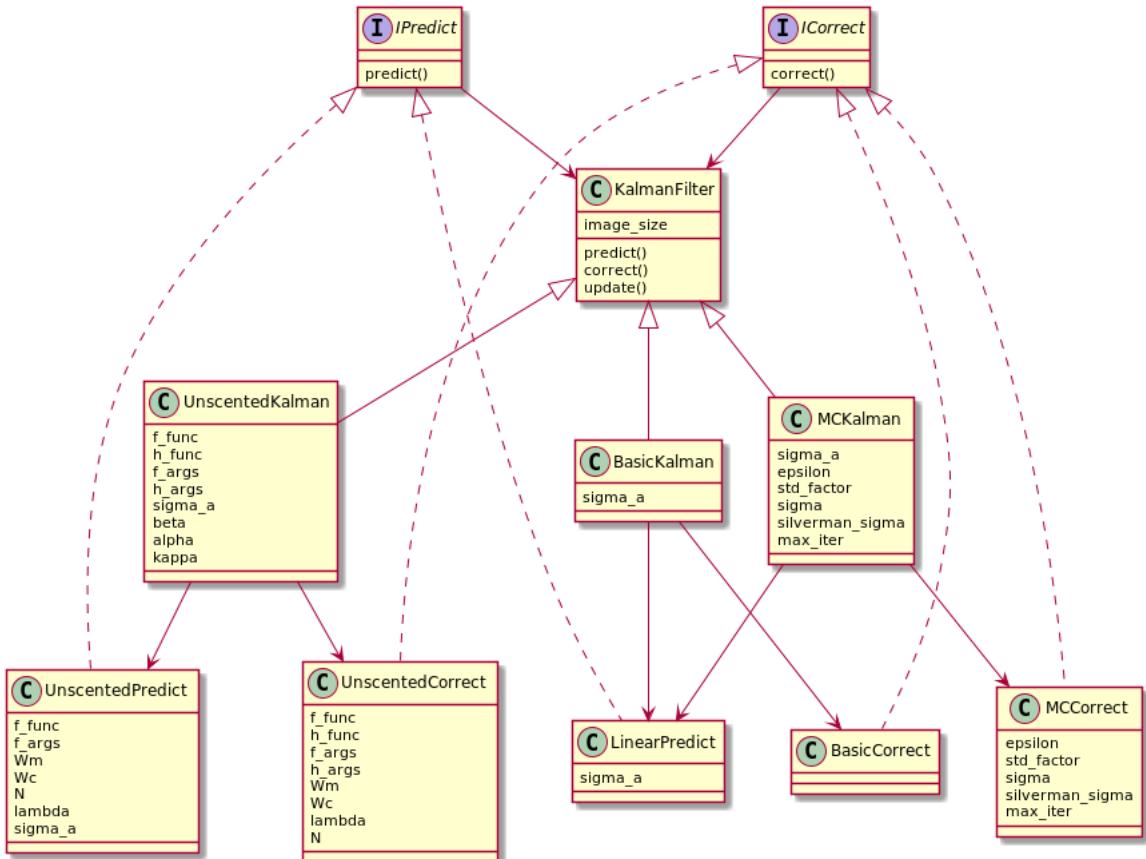


Figura 5.1: Diagrama de clases de la familia de filtros de Kalman resultante. El patrón Strategy permite que las funcionalidades de predicción y corrección puedan definirse en diferentes clases, facilitando la modularización y la implementación de nuevos filtros como el unscented.

6 | Resultados

En este capítulo se exponen resultados y análisis de diferentes pruebas realizadas con la nueva versión desarrollada del programa. La primera de las pruebas consiste en medir el desempeño de esta pipeline en términos de tiempo de ejecución y uso de memoria principal ocupada al procesar tres datasets (usados en el capítulo 3) asociados a supernovas encontradas por HiTS [8]: SN14, SN18 y SN80. El segundo conjunto de experimentos corresponde al procesamiento de los 93 conjuntos de datos asociados a las supernovas encontradas por HiTS en el semestre de 2015. En ambos tipos de pruebas se busca comparar el rendimiento y los resultados del programa al emplear los diferentes filtros de Kalman implementados. Finalmente se muestran algunos resultados específicos obtenidos con el filtro unscented.

Para todas las experiencias se empleó umbrales de flujo y velocidad de flujo estimada de 200 [ADU] y 50 [ADU/día]. Además todas las imágenes procesadas poseen la misma dimensión: 4094×2046 de alto por ancho, por lo que el campo `image_size` de todos los filtros se configuraron en (4094, 2046). Por cada filtro se emplearon los siguientes valores de parámetros:

Filtro de Kalman Básico

- `sigma_a`: Se ocupó un valor de 0,1. Este valor se ajustó después de realizar diferentes pruebas variando este parámetro entre 100 y 10^{-4} . Este término es empleado durante el proceso de predicción y también fue usado para ejecutar las pruebas con la versión anterior del programa.
- `init_var`: La varianza inicial de flujo y velocidad de flujo se estableció con un valor de 100,0. Tal como se empleó en el programa original.
- Se determinó 0,0 como valor inicial para las estimaciones de flujo y velocidad de flujo (estado). Fue el mismo valor usado para las pruebas realizadas con el programa original.

Filtro de Kalman de máxima correntropía

- `sigma_a`: Se empleó valor 0,1. Este valor es usado, al igual que en el caso del filtro básico, durante el proceso de predicción (mismo método).

- **sigma**: Se estableció un valor de 1000,0 para la desviación estándar por defecto, en caso de no usar optimización de Silverman durante proceso de corrección (valor determinado en la versión original del programa).
- **silverman**: **False**. Para estas pruebas no se usó el método de Silverman para obtener distribución ya que la implementación en el programa original se encuentra incompleta (por lo que una nueva implementación de esta función en el software desarrollado en este trabajo queda pendiente para trabajo futuro).
- **std_factor**: Se estableció como 100,0. Al no usarse el método de Silverman, tampoco se hace uso de este valor (valor establecido en la versión original del programa).
- **epsilon**: Se asignó valor de 10^{-6} , como error máximo. Es el mismo usado en la versión anterior del programa.
- **max_iter**: Se consideró 10 como número máximo de iteraciones, que es el número definido por defecto (en el diseño del programa original).
- **init_var** : La varianza inicial de flujo y velocidad de flujo se estableció con un valor de 100,0, tal como se hizo con la versión original.
- Se determinó 0,0 como valor inicial para las estimaciones de flujo y velocidad de flujo. Este valor se usó igualmente para las pruebas ejecutadas con el programa original.

Filtro de Kalman Unscented

- **f_func**: Se definió una función no lineal de grado 1,5 (información entregada en **f_args**).
- **h_func**: Se estableció como la función identidad.
- **f_args**: El grado de la función se definió como 1,5 y un factor de 1,0 (por tanto corresponde a la lista [1,5; 1,0]).
- **h_args**: Para estas pruebas se definió como una lista vacía ya que $h(\cdot)$ se trata de la función identidad.
- **alpha**: Se estableció como 0,001. Ya que con este valor se detectaron algunas supernovas de HiTS. A mayores valores disminuyen el número de candidatos encontrados (se probaron con valores de 0,0001 a 10).
- **beta**: Se le asignó un valor de 2,0. Se asumió distribución normal de las variables de estado [13].

- **kappa**: Se le asignó un valor de 0,0. Se asumió distribución normal de las variables de estado [13].
- **N**: Al tratarse de dos variables de estado, se establece N=2.
- **sigma_a**: Se definió como 0,1. Se consideró usar el mismo valor para este parámetro de acuerdo a los resultados obtenidos con los filtros básico y de máxima correntropía, ya que los resultados variaban negativamente (disminuyendo candidatos encontrados) al tomar valores (potencias de diez) mayores o menores a este número.

Las condiciones iniciales del filtro unscented fueron diferentes a las definidas para los filtros básico y de máxima correntropía, ya que se consideró la primera medición de flujo y su varianza para inicializar las matrices de estado y covarianza. Esto debido a que la función no lineal depende de un Δt medido desde cierta época t_0 . Para estos experimentos se definió t_0 como la primera época de observación (los otros métodos dependen de la diferencia temporal entre una época y otra). Por tanto, se configuró el flujo inicial al valor del flujo calculado de la primera época. Además, la varianza del estado del flujo fue definida como la matriz de varianza de flujo calculado.

6.1. Desempeño

En esta sección se describen los resultados obtenidos durante las experiencias de medición de desempeño de la nueva versión del programa en términos de tiempo y memoria principal usada.

Las pruebas se realizaron usando el mismo conjunto de datos utilizado para medir el desempeño del programa original (Capítulo 3), además se emplearon como herramientas de medición igualmente el perfilador de memoria `mem_profiler` y la librería RESOURCE¹. Del mismo modo en que se llevaron a cabo las pruebas usando el programa original, se empleó un computador personal de 8 GB de RAM, con un procesador de 2,8 GHz de frecuencia.

6.1.1. Tiempo de ejecución

Las Tablas 6.1, 6.2 y 6.3 muestran los resultados de tiempo medido en segundos de cada una de las partes del proceso de la rutina principal, para los tres datasets usando los filtros básico, de máxima correntropía y unscented, respectivamente.

A todos estos resultados se les ha agregado la medición de tiempo promedio por imagen (ya que las secuencias poseen largos diferentes). De estos resultados es posible deducir que el tiempo de cálculo de flujo por época siempre toma alrededor de 11,0 segundos. Esto es claro debido a

¹Método `get_rusage`

Tabla 6.1: Tiempo de ejecución en segundos de cada proceso involucrado, usando el filtro de Kalman básico refactorizado: cálculo de flujo, estimación de estados, detección de candidatos y guardado de candidatos resultantes en caso de haberlos. La última fila describe el tiempo promedio que toma por observación (en segundos igualmente) para cada uno de los procesos.

ID	Cálc. Flujos [s]	Estim. KF [s]	Detección [s]	Obt. Candidatos [s]
SN14	297,93	28,36	33,53	0,00
SN18	255,07	22,41	34,32	0,00
SN80	199,16	17,73	25,02	0,00
\bar{t}/Obs	11,20	1,02	1,39	0,00

Tabla 6.2: Tiempo de ejecución en segundos de cada proceso involucrado, usando el filtro de Kalman de máxima correntropía refactorizado: cálculo de flujo de las imágenes, estimación de estado, detección de fuentes y guardado de candidatos. La última fila describe el tiempo promedio que toma por observación (en segundos) para cada una de las tareas.

ID	Cálc. Flujos [s]	Estim. KF [s]	Detección [s]	Actual. Candidatos [s]
SN14	289,59	491,08	34,04	0,00
SN18	256,95	437,37	33,00	0,00
SN80	199,98	346,98	24,96	0,00
\bar{t}/Obs	11,00	19,06	1,38	0,00

Tabla 6.3: Tiempo de ejecución en segundos de cada tarea, usando el filtro de Kalman unscented: cálculo de flujo, estimación de filtros, detección de fuentes y guardado de candidatos. La última fila describe el tiempo promedio que toma por observación (en segundos) para cada uno de los procesos.

ID	Cálc. Flujos [s]	Estim. KF [s]	Detección [s]	Actual. Candidatos [s]
SN14	280,67	342,88	123,68	0,01
SN18	243,96	293,05	122,09	0,00
SN80	190,43	231,34	74,48	0,00
\bar{t}/Obs	10,65	12,93	4,73	0,00

que el proceso de cálculo de flujo es independiente del filtro escogido. Los tiempos de estimación de estado, por el contrario, varían de filtro en filtro. Tomando mayor tiempo al usar el filtro de máxima correntropía. Sin embargo, opuesto a lo que se esperaba (ya que el proceso de detección ocurre una vez realizada la estimación de estado) el filtro unscented presenta un tiempo de detección casi cuatro veces mayor a los tiempos medidos en los otros filtros; esto se podría deber a las operaciones lineales realizadas durante el proceso de corrección ya que existiría mayor consumo de RAM.

Con los filtros básico y de máxima correntropía no se encontraron candidatos en ninguno de los datasets. Esto probablemente puede deberse al cambio en el filtrado de las imágenes generado por el proceso de *refactoring*, siendo un proceso más riguroso en la nueva versión, algunas imágenes u archivos del dataset de SN14 fueron descartados. No ocurrió lo mismo con el filtro unscented al ejecutar la pipeline sobre los datos de la supernova SN14, ya que se encontraron candidatos. Sin

Tabla 6.4: Tiempo de exploración (para detección de candidatos) para cada uno de los filtros. La última fila corresponde a tiempo (en segundos) total promedio por observación. A la izquierda, tiempo empleado usando la nueva versión de filtro de Kalman básico. En el centro, los resultados de tiempo correspondientes al filtro de Kalman de máxima correntropía. A la derecha, resultados correspondientes al filtro unscented.

ID	Tiempo total [s]	ID	Tiempo total [s]	ID	Tiempo total [s]
SN14	359,82	SN14	814,71	SN14	747,24
SN18	311,8	SN18	727,32	SN18	659,1
SN80	241,91	SN80	571,92	SN80	496,25
$\bar{t}/Obs.$	13,61	$\bar{t}/Obs.$	31,58	$\bar{t}/Obs.$	28,32

embargo, al ser candidatos no confirmados por HiTS, pueden corresponder efectivamente otro tipo de objetos (no necesariamente supernovas), por lo que tendría que estudiarse su curva de luz o la curva generada por su espacio de estados (es decir, determinar índice de entropía de la curva).

La totalidad y el promedio del tiempo que toma la ejecución de la pipeline con los tres filtros sobre los datasets se muestran en la Tabla 6.4, en donde figuran los resultados de los filtros básico, de correntropía máxima y unscented, respectivamente.

De los resultados expuestos sobre el tiempo total de ejecución se desprende que la pipeline demora menos con el filtro básico, mientras que con el de máxima correntropía es con el toma mayor tiempo.

6.1.2. Uso de memoria

Las Figuras 6.1, 6.2 y 6.3 muestran el comportamiento del consumo de memoria (en mebibytes) en el tiempo (en segundos) al usar la nueva versión del filtro básico, de máxima correntropía y el nuevo filtro unscented, respectivamente, ejecutando la pipeline refactorizada sobre los tres conjuntos de datos (SN14, SN18 y SN80).

De la Figura 6.1 se desprenden los máximos alcanzados durante cada ejecución (en cada *dataset*) con el filtro básico. Los resultados en megabytes (MB) se listan en la Tabla 6.5, a la izquierda.

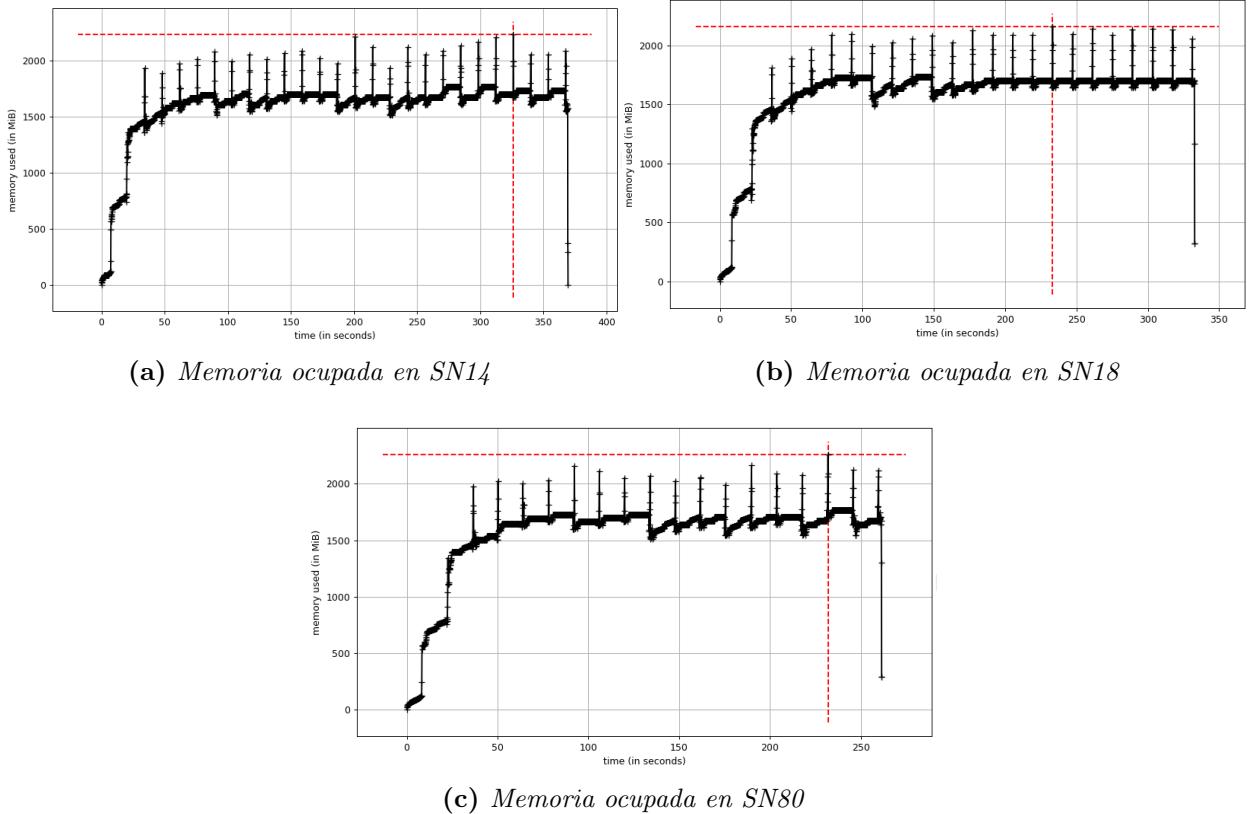


Figura 6.1: Comportamiento de la memoria (en mebibbytes) durante la ejecución para los tres conjuntos de datos usando el filtro de Kalman Básico.

De las series mostradas en la Figura 6.2 se obtienen los máximos alcanzados de memoria principal ocupada al utilizar el filtro de máxima correntropía listados en la Tabla 6.5 (centro).

Los resultados de memoria principal muestran un uso intensivo de esta al usar el filtro unscented (ver Tabla 6.5, derecha). Esto puede ser debido a la serie de operaciones lineales que se debe realizar al momento de predecir y corregir. También se desprende que el uso de memoria es menor con el filtro básico.

Tabla 6.5: Comparación de la memoria principal (en unidades de MB) usada durante la ejecución del programa para los tres filtros. A la izquierda, los resultados correspondientes al usar filtro de Kalman Básico. En el centro, memoria ocupada usando el filtro de Kalman de máxima correntropía. A la derecha, memoria ocupada usando filtro de Kalman unscented.

ID	Memoria [MB]
SN14	2347,78
SN18	2254,93
SN80	2365,07

ID	Memoria [MB]
SN14	3359,54
SN18	3350,53
SN80	3330,93

ID	Memoria [MB]
SN14	4827,82
SN18	4835,96
SN80	4821,89

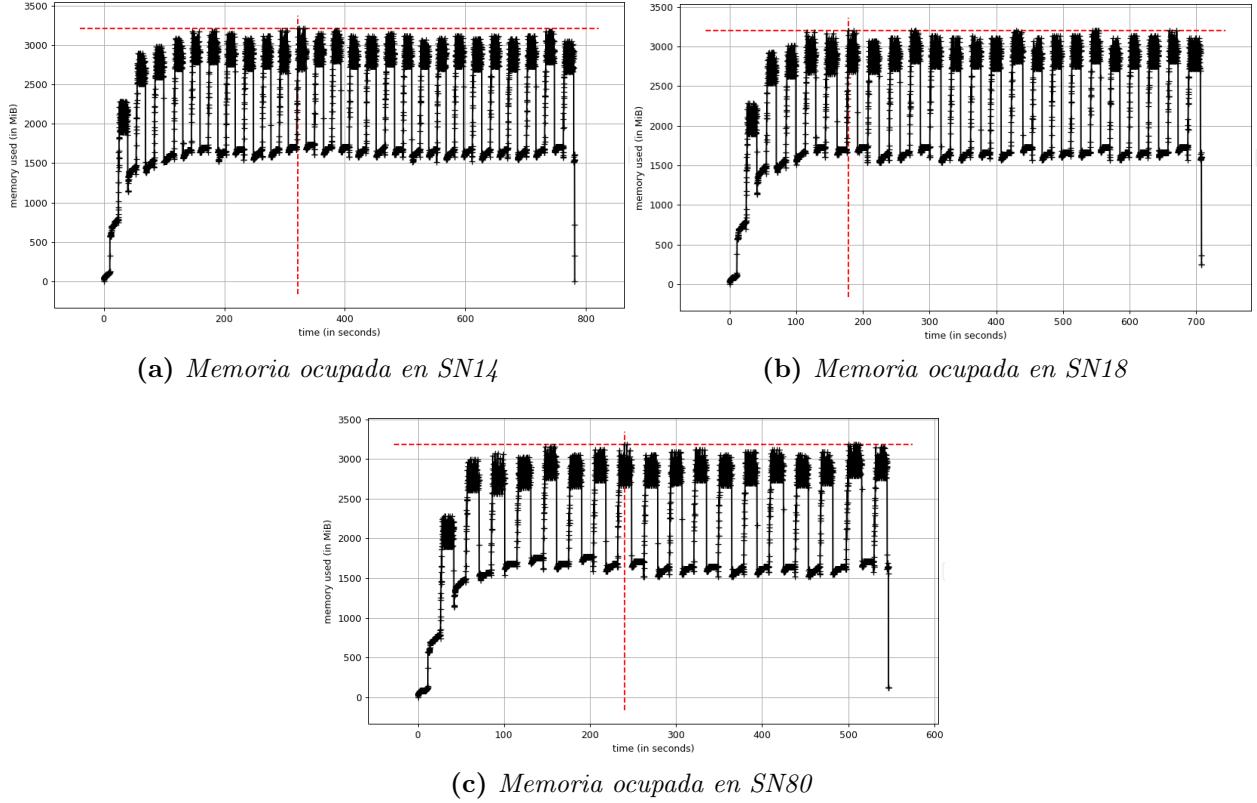


Figura 6.2: Comportamiento de la memoria (en mebibbytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman de máxima correntropía.

6.2. Detección

En esta sección se exponen los resultados obtenidos al procesar los datos de las 93 supernovas con los tres filtros implementados en Leftraru. En particular se muestran las cantidades de verdaderos positivos, falsos negativos, falsos positivos y el número de conjunto de datos que no pudieron ser procesados por no contar con algún tipo de imagen o archivo necesario para su procesamiento (en este caso, fue la falta de imágenes de diferencia). Se considera como falso positivo todo aquello que no esté catalogado como supernova de acuerdo a HiTS; es decir, estrellas variables u otro tipo de objetos celestes, rayos cósmicos, junto a ruido en la intensidad de los píxeles, etc.

Los parámetros usados por cada filtro son los mismos descritos al inicio de este capítulo.

6.2.1. Filtros básico y de máxima correntropía

La Tabla 6.6 muestra los resultados obtenidos con la nueva versión de la pipeline, empleando los filtros refactorizados básico y de máxima correntropía.

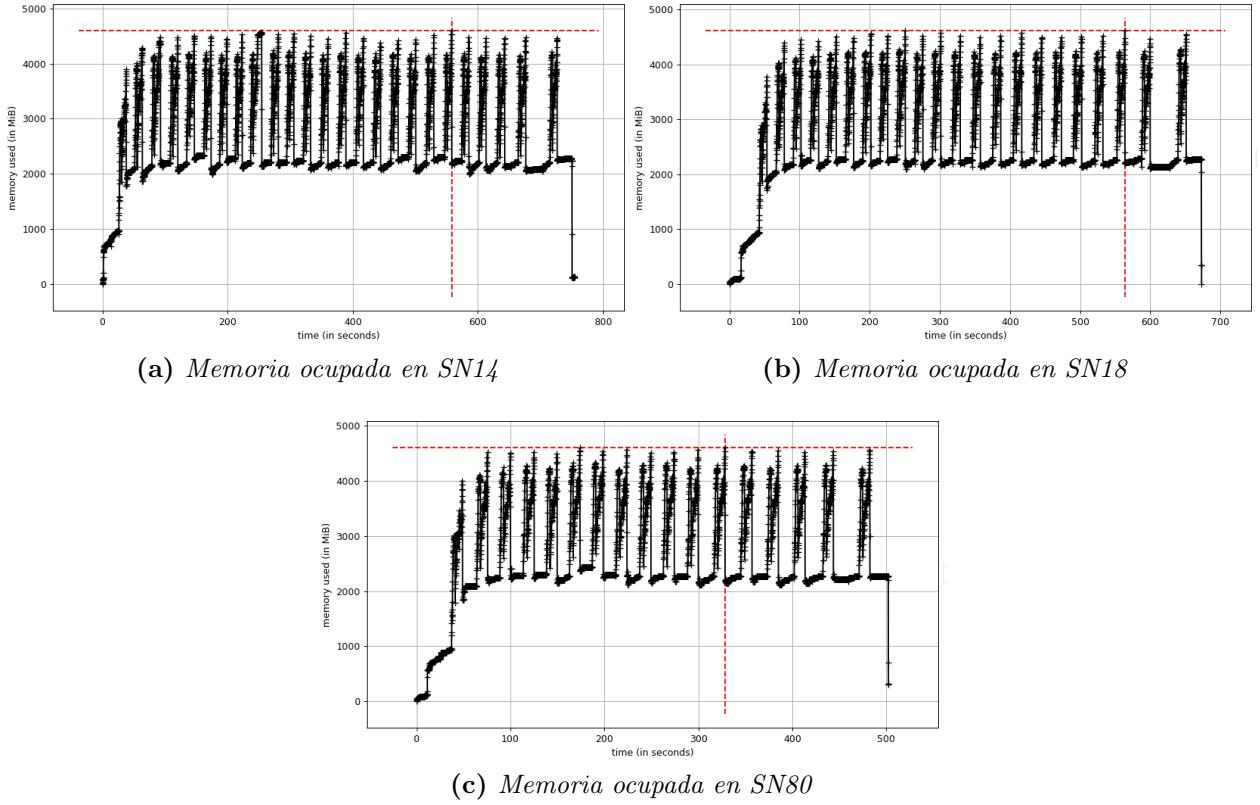


Figura 6.3: Comportamiento de la memoria (en mebibbytes) durante la ejecución para los tres conjuntos de datos. En los tres lanzamientos se usó el filtro de Kalman unscented.

Tabla 6.6: Número de verdaderos positivos (TP), falsos negativos (FN) y falsos positivos (FP) encontrados usando cada uno de los filtros en el conjunto de datos de HiTS. La cuarta columna de valores muestra la cantidad de conjuntos de datos que no pudieron ser procesados.

Filtro	TP	FN	FP	NaN
Básico	36	54	37	3
Máx. Corr.	36	54	40	3

De la Tabla 6.6 se obtiene que el filtro de máxima correntropía encuentra la misma cantidad de supernovas conocidas que el filtro básico, y detecta más falsos positivos. Sin embargo cabe recalcar que para esta experiencia no se usó el método de Silverman para estimar σ en el proceso de corrección, por lo que podría existir opciones de mejora.

De acuerdo a la Tabla 7.1, Apéndice K varias de las detecciones ocurren después del período de alta cadencia (después de MJD 57075,00) y prácticamente no hay diferencia entre ambos filtros. Lo primero puede deberse a que es necesario cumplir las condiciones mencionadas en el Apéndice G en los métodos de la clase SOURCEFINDER por cuatro épocas consecutivas para emitir una alerta. Por otro lado otras detecciones realizadas por ambos filtros ocurrieron en un período bastante alejado del intervalo de alta cadencia, por lo que el distanciamiento entre las observaciones puede haber

favorecido a la detección.

6.2.2. Filtro unscented

Con el filtro unscented se realizaron dos pruebas. En ambos casos se asumió la función $h(\cdot)$ como la función identidad. En cambio la función $f(\cdot)$ tomó dos formas no lineales en función de Δt : $f(\Delta t) = (\Delta t)^{1.5}$ y $f(\Delta t) = (\Delta t)^2$. Los resultados obtenidos se muestran en la Tabla 6.7.

Tabla 6.7: Número de verdaderos positivos (TP), falsos negativos (FN) y falsos positivos (FP) encontrados usando el filtro unscented para dos tipos de funciones no lineales: $f(\Delta t) = (\Delta t)^{1.5}$ y $f(\Delta t) = (\Delta t)^2$, sobre el dataset de HiTS. La tercera columna de valores muestra la cantidad de conjuntos de datos que no pudieron ser procesados.

Filtro	TP	FN	FP	NaN
$\Delta t^{1.5}$	2	87	110	3
Δt^2	3	87	108	3

Se observa que bajo la configuración seleccionada de valores de parámetros, el desempeño de la detección es bastante pobre. No obstante se debe considerar la posibilidad de mejorar este resultado estudiando condiciones iniciales, valores de parámetros y en particular, las funciones a utilizar ($f(\cdot)$ y $h(\cdot)$).

Para observar si el filtro unscented está trabajando como se espera, es posible visualizar que está sucediendo con las estimaciones y los valores de flujo con alguna de sus detecciones. Se seleccionó una de las supernovas encontradas por este filtro en el CCD N2, campo 7 usando una función $f(\Delta t) = \Delta t^2$.

De acuerdo a las Figuras 6.4, 6.5 y 6.6 uno de los problemas que adolece este filtro es el de crecimiento rápido, por lo que de alguna forma se debe contrarrestar esta gran alza trabajando, tal vez, sobre la función de predicción $f(\cdot)$.

6.3. Observaciones

Respecto de las detecciones obtenidas con los filtros básico y de máxima correntropía tanto para esta versión como para la original, se detectaron el mismo número de supernovas: treinta y seis. Sin embargo disminuyeron notablemente los falsos positivos respecto de la versión original y además se encontraron tres supernovas más de las confirmadas por HiTS. Esto último se piensa puede deberse a un nuevo ordenamiento y filtrado más precisos de los archivos necesarios para ejecutar la rutina.

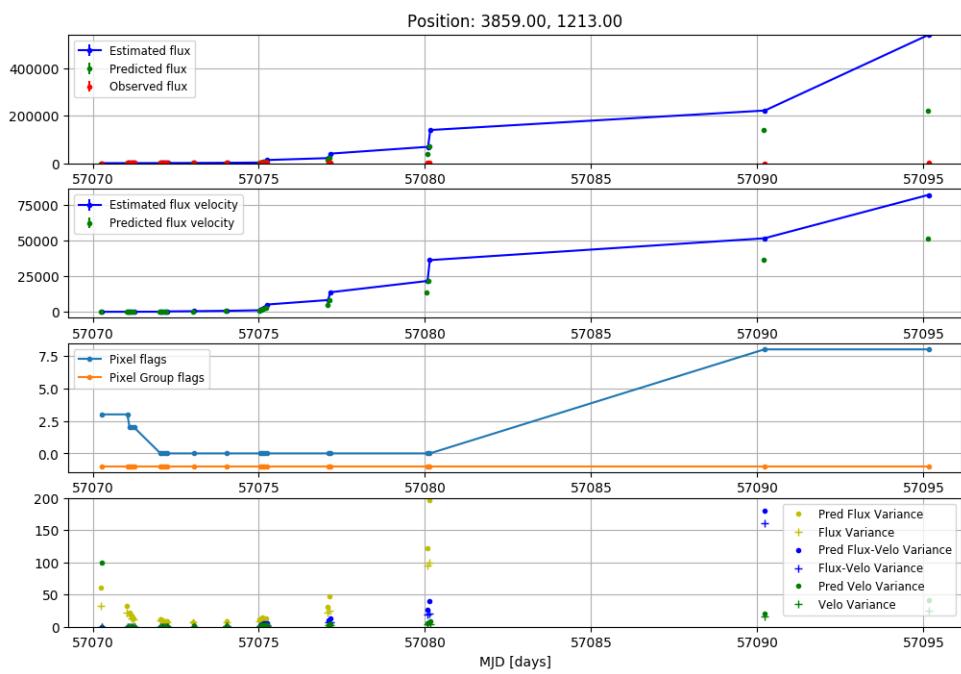


Figura 6.4: En la primera gráfica de arriba hacia abajo se muestra el comportamiento del flujo medido y el flujo predicho y estimado. En esta representación se observa que el flujo se dispara a partir de la fecha 57075. De la misma forma, se dispara las velocidades de flujo estimado y predicho de acuerdo al segundo gráfico. El siguiente esquema indica el crecimiento de *flags* de invalidez de los píxeles mientras que la última imagen da cuenta del rápido crecimiento de las varianzas y covarianzas de estimación y predicción de las variables de estado.

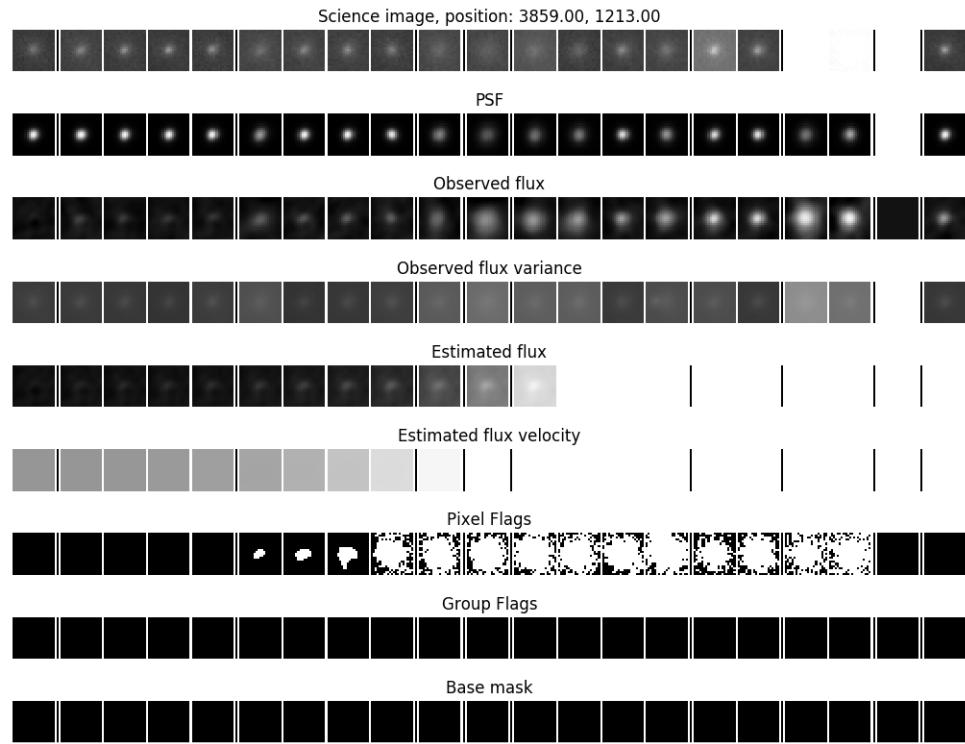


Figura 6.5: Representación en píxeles de lo que sucede con el flujo observado, las estimaciones y los *flags* de los píxeles. Se distingue la saturación sobre las matrices de flujo y velocidad estimada.

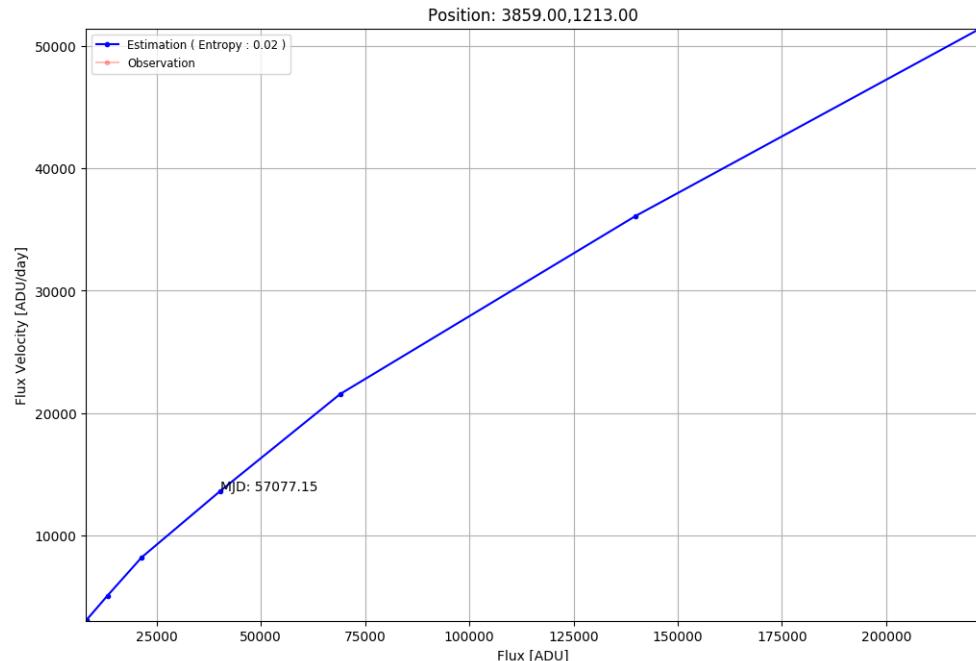


Figura 6.6: Se observa el comportamiento conjunto del flujo y velocidad estimados. El rápido crecimiento de ambas variables es bastante notable

Por otro lado en términos de desempeño, si bien la nueva versión ha podido disminuir tiempos de ejecución, aún no se logra disminuir la cantidad de memoria principal usada respecto del programa antecesor al emplear los filtros básico y de máxima correntropía.

7 | Conclusión

Se propuso una nueva versión optimizada del software implementado originalmente por Pablo Huentelemu destinado a la detección de supernovas, obteniéndose también una familia de filtros de Kalman para el análisis de imágenes astronómicas que puede ser fácilmente extendida agregando nuevos métodos de estimación basados en el filtro de Kalman. Esta extensión es facilitada gracias al patrón Strategy empleado en el diseño.

Por otro lado, otro de los beneficios adquiridos en esta nueva versión es el control sobre los argumentos de entrada del programa. Anteriormente, las expresiones regulares de los nombres de los archivos tenían que ser escritos explícitamente en el código del programa. Sin embargo el código rediseñado en este trabajo permite la recepción de estas expresiones regulares en un archivo de texto plano por lo que el usuario sólo debe entregar las rutas y expresiones regulares correspondientes de las imágenes y archivos en este documento. Además, cabe destacar que igualmente se independizaron del código los valores de entrada de umbrales y de parámetros requeridos en las diferentes rutinas entregándose de la misma forma en un archivo de texto. Es decir se logró evitar el *hard-coding* presente en la versión anterior de la pipeline.

Se debe agregar, que dentro de los cambios agregados a la nueva versión de la pipeline se acortó tiempo del proceso ya que a priori se asume que no se conoce ninguna supernova, es decir, el programa no tiene porqué saber si existe un candidato conocido, por lo que los resultados de los potenciales aspirantes a supernova son tratados ecuánimamente, y por tanto no se discrimina en la información obtenida en los resultados al ser guardada en disco. Por ende, el análisis se realiza en una pasada (lo que le permite demorarse la mitad del tiempo original) y no en dos como se realizaba antiguamente.

Respecto de las pruebas realizadas en la evaluación de las versiones original y nueva, se observa que efectivamente la nueva versión de los filtros básico y de máxima correntropía logran detectar más supernovas que sus respectivas versiones originales (considerando el número de datasets sobre los cuales se trabajó), además de reducir el número de falsos positivos para el mismo conjunto de

umbrales. Este resultado se presume que puede ser debido a la mejora en el ordenamiento de los datos de entrada ya que este es aplicado sobre todos los datasets de entrada, tanto a imágenes FITS como a archivos de extensión NPY necesarios para la rutina y no sólo sobre las primeras como lo realiza el programa original. Del análisis de desempeño se desprende que entre las versiones antigua y refactorizada se observa que hay una mejora los tiempos de procesamiento. Sin embargo se mantiene el uso de memoria principal, tanto para el filtro básico como el de máxima correntropía.

Se refactorizó igualmente el proceso de visualización de resultados; con esta nueva versión no sólo es posible observar las series de tiempo de la evolución de las mediciones y observar el comportamiento de los píxeles, flujos y estimaciones gráficamente (como estampillas) sino además se puede visualizar el resultado de las estimaciones de estado para algún candidato específico en el espacio de fase, indicando fecha de detección (como MJD) y el valor de la complejidad de la curva generada en términos de su entropía.

Se implementó además un nuevo miembro de la familia de filtros de Kalman el cuál permite realizar aproximaciones con funciones que dependan del paso del tiempo respecto de algún instante de referencia particular. Sin embargo los primeros resultados de este filtro dan cuenta de que se hace necesario estudiar y probar nuevos conjuntos de parámetros, condiciones iniciales y funciones, ya que para este trabajo sólo se utilizó un rango acotado de estos elementos. Igualmente faltó explorar el comportamiento del filtro de máxima correntropía usando el método de Silverman.

Por otra parte se estima que el hecho de que los datos usados para la realización de este trabajo presenten un muestreo irregular complica la obtención de buenos resultados en el proceso de detección ya que durante la corrección del filtro de Kalman, se requiere una observación (medición) para realizar las estimación y por tanto, afecta la siguiente predicción aumentando su incertezza si entre observaciones ha transcurrido demasiado tiempo.

7.1. Trabajo futuro

Uno de los aspectos pendientes y que se esperaría trabajar en el futuro es el estudiar la variación de resultados del filtro unscented usando diferentes funciones no lineales. Además investigar que sucede al variar los parámetros como σ_a . De la misma forma sería deseable explorar lo que ocurre con el filtro de máxima correntropía al momento de usar el método de Silverman.

Por otra parte queda también pendiente, el estudiar el comportamiento de los resultados al variar los umbrales relacionados con la estimación realizada por el filtro de Kalman y que son usados en la fase de reconocimiento de candidatos (en la clase SOURCEFINDER).

Los métodos presentados podrían ser usados en la detección de estrellas variables aplicando un filtro que distinga tendencias decrecientes en la luminosidad de estos objetos; por lo que una extensión prometedora de este sistema podría diseñarse para reconocer alternancia en los regímenes creciente y decreciente, haciendo uso de la estructura de clases dada por el patrón Strategy en el modelo del filtro.

Glosario

Airmass Es el largo del camino de que le toma a los rayos de una cuerda celeste atravesar la atmósfera. A medida que los rayos van penetrando la atmósfera estos se van atenuando por la absorción y el proceso conocido como scattering. La cantidad de aire directamente perpendicular a la tierra se conoce como *un airmass* y varía con la elevación sobre el nivel del mar.

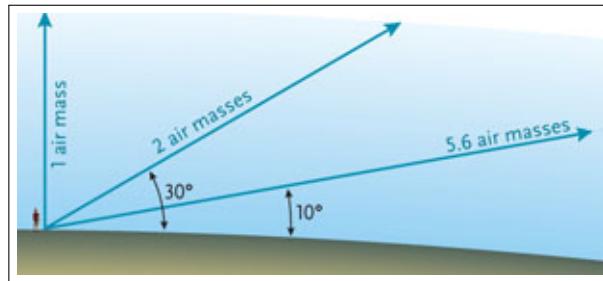


Figura 7.1: Mientras más cerca esté el objeto del horizonte, mayor será la cantidad de airmass por la que se observará y mayor será la distorsión sobre la imagen percibida del objeto. *Reed, C. 2008; <http://www.skyandtelescope.com/astronomy-resources/transparency-and-atmospheric-extinction/>*

Kernel En el procesamiento de imágenes, un kernel corresponde a una matriz convolucional. Por lo general es usada para la modificación de imágenes o detección de bordes. Un ejemplo de kernel para este caso es el kernel PSF.

En estadística bayesiana, el kernel de una densidad de probabilidad es la forma de esta en la que se ha omitido todo tipo de factor que no es función de ninguna de las variables del dominio.

PSF Corresponde a la respuesta instrumental a una fuente de luz puntual, cuya radiación debe atravesar la atmósfera terrestre y los lentes del telescopio. La distorsión puede ser interpretada como la convolución de la imagen por un kernel.

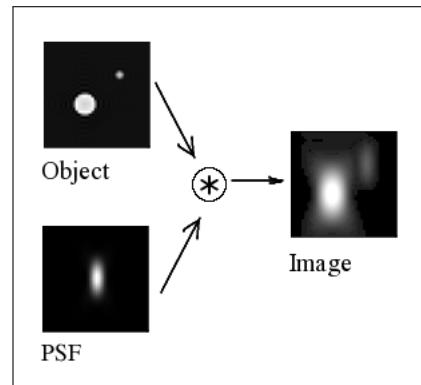


Figura 7.2: Ejemplo de distorsión de una fuente al aplicar un kernel de PSF específico. El resultado se observa en el cuadro *Image*. *P. Huentelemu, 2016 [15]*

Bibliografía

- [1] H. Aihara, N. Arimoto, R. Armstrong, S. Arnouts, N. A. Bahcall, S. Bickerton, J. Bosch, K. Bundy, P. L. Capak, J. H. Chan, et al. The hyper suprime-cam ssp survey: overview and survey design. *Publications of the Astronomical Society of Japan*, 70(SP1):S4, 2017.
- [2] M. Arnaboldi, M. Capaccioli, D. Mancini, R. Scaramella, G. Sedmak, and R. Kurz. The vst-vlt survey telescope. In *From Extrasolar Planets to Cosmology: The VLT Opening Symposium*, pages 204–208. Springer, 2000.
- [3] A. Balestrino, A. Caiti, and E. Crisostomi. Generalised entropy of curves for the analysis and classification of dynamical systems. *Entropy*, 11(2):249–270, 2009.
- [4] R. D. Blum, K. Burleigh, A. Dey, D. J. Schlegel, A. M. Meisner, M. Levi, A. D. Myers, D. Lang, J. Moustakas, A. Patej, et al. The decam legacy survey. In *American Astronomical Society Meeting Abstracts # 228*, volume 228, 2016.
- [5] B. Chen, X. Liu, H. Zhao, and J. C. Príncipe. Maximum Correntropy Kalman Filter. *ArXiv e-prints*, Sept. 2015.
- [6] B. Chen, X. Liu, H. Zhao, and J. C. Príncipe. Maximum correntropy kalman filter. *Automatica*, 76:70 – 77, 2017.
- [7] J. Emerson, W. Sutherland, A. McPherson, S. Craig, G. Dalton, and A. Ward. The visible & infrared survey telescope for astronomy. *The Messenger*, 117:27–32, 2004.
- [8] F. Forster, J. C. Maureira, S. Gonzalez-Gaitan, G. Medina, G. Pignata, L. Galbany, J. San Martin, M. Hamuy, P. Estevez, R. C. Smith, K. Vivas, S. Flores, P. Huijse, G. Cabrera, J. Anderson, F. Bufano, T. de Jaeger, J. Martinez, R. Munoz, E. Vera, and C. Perez. HiTS real-time supernova detections. *The Astronomer's Telegram*, 7146, Feb. 2015.
- [9] B. J Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Phil. Trans. R. Soc. Lond. A*, 209(441-458):415–446, 1909.

- [10] N. Kaiser, H. Aussel, B. E. Burke, H. Boesgaard, K. Chambers, M. R. Chun, J. N. Heasley, K.-W. Hodapp, B. Hunt, R. Jedicke, et al. Pan-starrs: a large synoptic survey telescope array. In *Survey and Other Telescope Technologies and Discoveries*, volume 4836, pages 154–165. International Society for Optics and Photonics, 2002.
- [11] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME - Journal of Basic Engineering*, 82:35–45, 1960.
- [12] A. P. A. LSST Science Collaboration, A. J., and A. S. F. et al. *LSST Science Book, Version 2.0*. arXiv:0912.0201, 2009.
- [13] E. Matsinos. The Kalman Filter: a didactical overview. *ArXiv e-prints*, July 2016.
- [14] T. Naylor. An optimal extraction algorithm for imaging photometry. *Monthly Notices of the Royal Astronomical Society*, 296(2):339–346, 1998.
- [15] H. P. Filtro de Correntropía para detección de supernovas. *Tesis de magíster en ciencias de la ingeniería, mención eléctrica de la Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas*, 2016.
- [16] I. Reid. Estimation ii. *Oxford University, Department of Engineering Science*, 2001.
- [17] E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158. Ieee, 2000.
- [18] E. A. Wan and R. Van Der Merwe. The unscented kalman filter. *Kalman filtering and neural networks*, pages 221–280, 2001.

Apéndices

A . Librerías usadas para el refactoring

Versión de Python: 3.6

- **pandas: 0.24.4**
- **matplotlib: 2.2.2**
- **numpy: 1.13.3**
- **mahotas: 1.4.4**
- **astropy: 3.0.2**

B . Rutas a directorios y expresiones regulares de archivos

Los campos se describen a continuación:

- **maskDir**: Directorio donde se almacenan las imágenes máscara (imágenes que identifican píxeles que no deben ser considerados).
- **scienceDir**: Directorio donde se almacenan las imágenes científicas (imágenes base ya pre-procesadas).
- **diffDir**: Directorio donde se almacenan las imágenes de diferencia (resta entre las imágenes base y científica).
- **psfDir**: Directorio donde se encuentran los modelos de psf usados para la determinación del flujo.
- **invDir**: Directorio que guarda las imágenes correspondientes a la varianza inversa (*peso* de cada pixel en términos de ruido: a menor peso, mayor ruido).
- **afluxDir**: Directorio que contiene los archivos de extensión NPY dentro de los cuales se guarda el valor de la variable **aflux**.
- **maskRegEx**: Expresión regular con la que es posible identificar el nombre de las imágenes máscara en disco siguiendo el path **maskDir**.
- **scienceRegEx**: Expresión regular con la que es posible identificar el nombre de las imágenes científicas en disco siguiendo el path **scienceDir**.
- **diffRegEx**: Expresión regular con la que se identifican el nombre de las imágenes de diferencia en disco siguiendo el path **diffDir**.
- **invRegEx**: Expresión regular con la que es posible identificar el nombre de las imágenes de la varianza inversa siguiendo el path **invDir**.
- **afluxRegEx**: Expresión regular con la que se identifica el nombre de los archivos *match* que contienen el valor de **aflux**. Estos archivos están ubicados en el path **afluxDir**.
- **psfRegEx**: Expresión regular que describe el nombre de las imágenes que guardan el modelo de PSF en el directorio **psfDir**.

B .1. Archivo de entrada: configuración de paths

```
maskDir = /home/apps/astro/data/DATA/Blind%s_%s/%s
scienceDir = /home/apps/astro/data/DATA/Blind%s_%s/%s
diffDir = /home/apps/astro/data/DATA/Blind%s_%s/%s
invDir = /home/apps/astro/data/DATA/Blind%s_%s/%s
afluxDir = /home/apps/astro/data/SHARED/Blind%s_%s/%s/CALIBRATIONS
psfDir = /home/apps/astro/data/SHARED/Blind%s_%s/%s/CALIBRATIONS
maskRegEx = Blind%s_%s_%s_[0-9][0-9]_dqmask.fits.fz
scienceRegEx = Blind%s_%s_%s_[0-9][0-9]_image_crblaster_grid02_lanczos2.fits
diffRegEx = Diff_Blind%s_%s_%s_[0-9][0-9]t*-0[0-9]t*_grid02_lanczos2.fits
invRegEx = invVAR_Blind%s_%s_%s_[0-9][0-9]t*-0[0-9]t*_grid02_lanczos2.fits
afluxRegEx = match_Blind%s_%s_%s_[0-9][0-9]-0[0-9].npy
psfRegEx = psf_Blind%s_%s_%s_[0-9][0-9]t*-0[0-9]t*_grid0[1-2]_lanczos2.npy
```

C . Diccionario de parámetros y umbrales

- **imgHeight**: Altura de las imágenes científicas en píxeles. Esta dimensión se propaga al resto de imágenes y matrices. Las imágenes usadas en este trabajo poseen una altura de 4094 píxeles.
- **imgWidth**: Ancho de las imágenes científicas en píxeles. Esta dimensión se propaga al resto de imágenes y matrices, y en este trabajo las respectivas imágenes poseen un ancho de 2046.
- **filter**: Tipo de filtro (**basic**, **mcc** o **ukf**).
- **results**: Directorio de resultados (donde se guardan las coordenadas de los candidatos encontrados junto a lista de épocas en que fueron detectados) en formato NPZ.
- **init_var**: Varianza inicial que tendrán las matrices de covarianza durante el proceso de estimación con los filtros de Kalman.
- **flux_thresh**: Umbral para estado de flujo obtenido con Kalman. Es un valor determinado por el usuario (en este trabajo se definió como 200).
- **flux_rate_thresh**: Umbral para la velocidad de flujo obtenido con Kalman. Es un valor establecido por el usuario (en este trabajo se definió como 50).
- **rate_satu**: Tasa de saturación en la velocidad de flujo. Para la realización de este trabajo se fijó como 3000, a partir de pruebas en las que se buscaba minimizar los falsos positivos.
- **sigma_a**: Varianza de la distribución de la componente de control (u_k) asumiendo normalidad. Es importante al emplear los filtros básico y unscented, ya que corresponde a la desviación estándar de la distribución normal de la aceleración originada por cambios no esperados en el modelo (lo que se puede interpretar como “fuerzas externas”) [16].
- **epsilon**: Radio de error con que la estimación por filtro de Kalman de máxima correntropía disminuye la ganancia de Kalman. Corresponde a un criterio de detención y toma valores entre 0 y 1 (típicamente, 10^{-6}) [5].
- **max_iter**: Número de iteraciones máximo para el proceso de corrección al usar Kalman de máxima correntropía.
- **silverman**: *boolean*. Toma valor 1 (*true*) en caso de considerarse, y 0 (*false*) si no. Se establece si se usa o no la aproximación de Silverman para determinar ancho de banda del kernel al emplear el filtro de máxima correntropía.

- **std_factor**: Factor de incremento de sigma al usar el método de Silverman.
- **sigma**: Ancho de banda usado por defecto sin Silverman en la determinación del kernel durante el proceso de corrección con el Filtro de Kalman de correntropía máxima .
- **beta**: Parámetro relacionado con la distribución del estado estimado (x_k). Toma valor de $\beta = 2$ para distribuciones normales.
- **kappa**: Participa en la regulación del rango de los valores de los puntos sigma y usualmente toma valores entre 1 y $3N-1$ (N corresponde al número de dimensiones) [18]. Aporta incremento adicional (ver Ecuación 2.54).
- **alpha**: Participa en la regulación del rango de los valores de los puntos sigma, y generalmente toma valores positivos menores o iguales a 1: $10^{-4} \leq \alpha \leq 1$ [18]. Incrementa el rango en un factor α (ver Ecuación 2.54).
- **dim**: Cantidad de componentes de estado a medir (en este programa se miden dos: flujo y su velocidad).

C .1. Archivo de entrada: parámetro y umbrales

```

imgHeight=4094
imgWidth=2046
filter=basic
results=/path/to/RESULTS/DIRECTORY/
plots=/path/to/PLOTS/DIRECTORY/
init_var=100.0
flux_thresh=200.0
flux_rate_thresh=50.0
rate_satu=3000.0
sigma_a=0.1
sigma=1000.0
epsilon=1e-6
max_iter=10
std_factor=100.0
silverman=0
beta=2.0
kappa=0.0
dim=2
alpha=0.001

```

D . Archivo de entrada: lista de campos, CCDs y semestres (incluye algunas coordenadas)

Field,CCD,Semester,POS_Y,POS_X
14,N16,15A,--,--
35,S21,15A,--,--
40,N10,15A,--,--
40,N17,15A,2837,1635
02,N5,15A,2755,805
07,N14,15A,2994,1528
47,N7,15A,--,--
07,N2,15A,--,--
45,N20,15A,3873,212
25,S14,15A,1111,1660
33,N28,15A,--,--
34,N27,15A,774,1246
13,S1,15A,--,--
41,N9,15A,3247,1851
26,S9,15A,4019,508
07,S10,15A,3300,518
26,N9,15A,895,988
22,N27,15A,--,--
20,N28,15A,3067,703
31,S4,15A,733,923
35,N23,15A,--,--
35,N5,15A,2801,1842
06,S23,15A,470,1885
05,N2,15A,--,--
13,N6,15A,617,773
08,S29,15A,480,362
21,S5,15A,3584,276
19,S2,15A,931,699
30,S21,15A,2818,1701
31,S2,15A,3559,1448
42,S17,15A,--,--
38,N21,15A,3491,1526
43,N17,15A,--,--
45,N25,15A,1041,1096
42,N31,15A,2908,1586

E . Métodos de la clase RoutineHandler

- `process_settings()`:

En este método se lee el archivo de diccionario de umbrales y parámetros con los que se configurará la toma de decisiones del programa.

- `retrieve_kalman_filter(kalman_string)`:

Corresponde a un método auxiliar que es invocado desde `process_settings` con el que se crea una instancia del filtro de Kalman a partir de la lectura del archivo de valores `settings_file`. Los tres strings válidos para la construcción de una instancia son: ‘basic’, ‘mcc’ y ‘ukf’. Si se entrega otro tipo de string, se levanta un error.

- `iterate_over_sequences(check_found_objects)`:

Recorre la lista de campos, CCDs y semestres entregada al programa con la consiguiente llamada a `routine`. Recibe como parámetro el argumento `check_found_objects` con el cual se indica si se quiere analizar resultados obtenidos anteriormente (candidatos encontrados), y que es entregado al método `routine` descrito a continuación.

- `routine(semester, field, ccd, results_path, check_found_objects, last_mjd)`:

Comprende la rutina principal del programa, es decir, el análisis de las observaciones de un semestre, campo y CCD específico. El argumento `check_found_objects` es un boolean e indicará el modo de ejecución del método: si es falso, sólo guardará las coordenadas de los candidatos encontrados (si no encontró nada, entonces se guarda una lista vacía) además de las épocas en que fueron detectados. Esta información se guarda en un arreglo de diccionarios. Si `check_found_objects` es verdadero, entonces cargará resultados anteriores del directorio de resultados (configurado en `process_settings`) para estudiar la presencia de los candidatos encontrados en caso de existir.

F . Métodos de la clase DataPicker

- `config_reg_expressions(semester, field, ccd)`

Este método recibe como parámetros strings que indiquen el semestre (`semester`), el campo (`field`) y el ccd (`ccd`) que se quiere analizar. Puede hacerse uso de los valores de las variables de instancia que la misma clase DATA_PICKER recibe en su constructor. Con estos strings se establecen las rutas de los directorios de las imágenes y las expresiones regulares de los nombres de las mismas.

- `collect_data()`

Esta función se encarga de recolectar la ruta completa de las diferentes imágenes (máscaras, imágenes científicas, de diferencia, etc.). Para esta finalidad se hace uso del método `walking_through_files`.

- `walking_through_files(regex, dir)`

Método con el cual se recorren las rutas definidas en los pasos anteriores y se agrupan los nombres completos (directorio incluido) de las imágenes ubicadas en el directorio `dir` y posean un nombre de patrón que siga la expresión regular `regex`.

- `filter_science_images()`

Filtrá imágenes científicas de acuerdo a su airmass, seleccionando aquellas obtenidas en fechas cuyo valor es menor a 1,7. De esta secuencia de imágenes científicas resultante se obtiene una lista de fechas que cumplen esta condición, medidas en términos de *día juliano modificado* o *Modified Julian Date* (MJD). Estos valores, de tipo punto flotante, son ordenados de forma creciente.

- `select_fits(dir)`

Selecciona y ordena los elementos de la lista de imágenes de formato FITS del directorio `dir` usando la lista de MJDs (guardada en la variable de instancia `mjd` de la clase) como resultado de `filter_science_images()` escogiendo sólo aquellas imágenes cuyas fechas correspondan a las fechas indicadas.

- `select_npys(dir, ref_dir, init_index, n_pos, rest_len):`

Debido a que los archivos de extensión NPY no poseen la variable MJD en su estructura

(en los archivos FITS encontramos este valor en el header de la imagen) deben de filtrarse de forma diferente. Para este caso el filtrado de este tipo de archivos se lleva a cabo a través de la revisión de sus nombres, ya que comparten patrones con los nombres de ciertas imágenes FITS. Por ejemplo, los nombres de las imágenes de PSF, de formato NPY, poseen similitud con los nombres de las imágenes FITS de diferencia; igualmente los archivos `aflux` de formato NPY poseen parecidos en sus nombres con las imágenes científicas. Esta similitud es medida a través de un substring diferente para cada tipo de archivo NPY, definido por la posición inicial `init_index`, en el nombre del archivo FITS y largo `rest_len`. `n_pos` indica la posición de un carácter específico ‘_’ en dicho substring para validar esta comparación.

G . Métodos de la clase SourceFinder

- **pixel_discard():**

Método en el que se realiza el descarte de píxeles de forma individual, siguiendo los siguientes criterios:

1. Si el flujo estimado por el filtro de Kalman para un pixel es menor que el umbral dado.
2. Si la velocidad de flujo estimada por el filtro de Kalman es menor que el umbral de la velocidad de flujo multiplicado por la tasa de saturación en la velocidad de flujo.
3. Si un pixel de la imagen científica es menor a la mediana más cierto delta (en este trabajo, siguiendo la línea de desarrollo de P. Huentelemu [15], se consideró 5.0) es descartado.
4. Si las varianzas de flujo son mayores a 150.0 (valor estimado por el autor del software original [15]).
5. Si las varianzas de la tasa de cambio de flujo (o velocidad de flujo) es mayor o igual a 150.0.
6. Si los píxeles no caen en etiquetas de invalidación dentro de la máscara que ha sido procesada para marcar también los píxeles vecinos a los realmente defectuosos.
7. Si los píxeles no han caído dentro del descarte por superar la mediana estimada a partir de cuatro épocas.

- **grouping_pixels():**

Este método trabaja con las etiquetas determinadas con el método anterior, en un arreglo de matrices (`numpy array`) denominado `pixel_flags` (variable de instancia). Además recibe el índice de MJD correspondiente a la observación de tal fecha. La agrupación de píxeles se realiza gracias a funciones brindadas por la librería Mahotas, usando el método `label` para encontrar dominios cerrados en el mapa de píxeles validados.

- **filter_groups(science, flux, var_flux, state, base_mask):**

Este método recibe la imagen científica, el flujo y su varianza, el estado determinado por el filtro de Kalman y la máscara correspondiente a una época específica. El filtrado de grupos de píxeles se lleva a cabo bajo las siguientes reglas de descarte:

1. Descarte de grupo por contener posible mala resta alrededor (valores negativos).
2. Si no hay máximos locales dentro del grupo de píxeles encontrados dentro de la imagen científica.

3. Si no hay máximos locales dentro del grupo de píxeles encontrados en la matriz de flujo (calculado por `calc_fluxes`).
4. Si no hay máximos locales dentro del grupo de píxeles encontrados en la matriz velocidad de flujo.
5. Si los valores de los píxeles superan la mediana local en imagen científica.
6. Si el grupo posee algún pixel que doble el valor del flujo o de la imagen científica.
7. Si el centro del grupo se encuentra etiquetado como defectuoso dentro de la máscara.
8. Si el pixel del centro del grupo se encuentra rechazado al ser superior a la mediana de los píxeles de cuatro observaciones consecutivas.
9. Si la varianza del flujo del pixel del centro del grupo es mayor al determinado por el umbral.

- `update_candidates(mjd)`:

En la estructura `CandData` se van registrando fechas (MJD) en que se han detectado candidatos previamente o se han detectado por primera vez. Es una estructura tipo lista en la que se van guardando diccionarios que contienen, cada uno, las coordenadas de un objeto, las épocas en las que ha sido detectado y si corresponde o no a una supernova conocida.

- `check_candidates(SN_index, SN_pos)`:

Verifica que dado un índice de supernova (`SN_index`) y sus coordenadas `SN_pos` se ha detectado dentro de los candidatos encontrados.

- `draw_complying_pixel_groups(science, state, state_cov, base_mask, dil_mask, flux, var_flux, mjd)`:

Este método es el que llama a `pixel_discard` para etiquetar píxeles para el descarte y no ser considerados en el paso de agrupamiento al llamar a `grouping_pixels`. Luego se invoca el método `filter_groups` para hacer el descarte a nivel grupal y obtener candidatos. La última llamada es para el método `update_candidates` para actualizar lista de candidatos encontrados en la variable de instancia `CandData`.

Como argumentos recibe todos los elementos necesarios para ejecutar los métodos que llama.

H . Métodos en utils

- `naylor_photometry(invvar, diff, psf):`

Calcula el producto del flujo por su varianza. Retorna el producto y la varianza. Para esto obtiene el flujo a partir de la imagen PSF entregada (`psf`) y del producto entre la imagen diferencia y la de varianza inversa (`diff` y `invvar` respectivamente) [14].

- `calc_fluxes(diff, psf, invvar, aflux):`

Calcula el flujo y su varianza gestionando la entrada y la salida de `naylor_photometry(invvar, diff, psf)`. Los valores NaN son transformados a valor constante 0.001.

- `subsampled_median(image, image_size, sampling):`

Extrae un número de `sampling` muestras de la imagen `image` con las cuales se estima su mediana.

- `cholesky(P):`

Calcula la descomposición de Cholesky de una estructura matricial de profundidad 3, en particular para operar sobre covarianzas.

- `mask_and_dilation(mask_path):`

Aplica una función sobre la máscara (ubicada en la ruta `mask_path`) para amplificar la zona de invalidez en torno a los píxeles que han sido marcados como defectuosos.

I . Métodos en unscented_utils

Se desarrollaron diferentes funciones auxiliares para apoyar el cálculo de las matrices:

- **sigma_points(mean_, cov_, lambda_, N):**

Función con la cual se calculan los puntos sigma a partir de la media de las variables de estado, la covarianza, el valor de λ y el número de variables de estado, N. Utiliza para esta finalidad, la descomposición de Cholesky.

- **unscen_weights(kappa, alpha, beta, N):**

Método con el que se calculan los pesos a partir de los valores de κ , α , β y N (número de variables de estado).

- **perform(func, *args):**

Función auxiliar con la cual se recibe un puntero a otra función vectorial (destinada a ser aplicada sobre un conjunto de puntos sigma) y un número arbitrario de argumentos, dependiendo de la necesidad de la misma función.

- **propagate_func(func, Wm, Wc, Xs, *args, N):**

Función con la que se propaga la función `func` sobre el conjunto de puntos sigma `Xs` usando los pesos de media y covarianza `Wm` y `Wc`, respectivamente, además del número de variables de estado, N. Además, recibe el argumento `args` que corresponde a una tupla de entradas propias de la función `func`.

J . Detección usando pipeline original

Tabla 7.1: Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando los filtros implementados originalmente (básico y de correntropía máxima).

Índ.	Básico	MCC	Índ.	Básico	MCC
1	57072,19	57072,19	48	57080,17	57080,17
2	-	-	49	57090,24	57090,24
3	57075,15	57075,15	50	57075,21	57075,21
4	57075,21	57075,21	51	-	-
5	57072,24	57072,24	52	-	-
6	-	-	53	-	-
7	-	-	54	-	-
8	57075,10	57075,10	55	-	-
9	57075,22	57075,22	56	57075,11	57075,11
10	57077,11	57077,11	57	57095,20	57095,20
11	57075,20	57075,20	58	-	-
12	57072,21	57072,21	59	57095,20	57095,20
13	57077,15	57077,15	60	-	-
14	-	-	61	-	-
15	57077,11	57077,11	62	57095,16	57095,16
16	57077,09	57077,09	63	-	-
17	57077,11	57077,11	64	-	-
18	57090,23	57090,23	65	-	-
19	-	-	66	-	-
20	57077,12	57077,12	67	-	-
21	-	-	68	-	-
22	57077,12	57077,12	69	-	-
23	-	-	70	-	-
24	57077,08	57077,08	71	-	-
25	57075,25	57075,25	72	-	-
26	-	-	73	-	-
27	57077,17	57077,17	74	-	-
28	-	-	75	-	-
29	-	-	76	-	-
30	57072,35	57072,35	77	-	-
31	-	-	78	-	-
32	57075,21	57075,21	79	-	-
33	-	-	80	-	-
34	-	-	81	-	-
35	-	-	82	-	-
36	57080,11	57080,11	83	-	-
37	57075,21	57075,21	84	-	-
38	-	-	85	-	-
39	-	-	86	57080,20	57080,20
40	-	-	87	-	-
41	-	-	88	-	-
42	-	-	89	-	-
43	57090,22	57090,22	90	-	-
44	-	-	91	NaN	NaN
45	-	-	92	NaN	NaN
46	-	-	93	NaN	NaN
47	57075,20	57075,20			

K . Detección usando pipeline refactorizada

Tabla 7.2: Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando los filtros refactorizados (básico y de correntropía máxima).

Índ.	Básico	MCC	Índ.	Básico	MCC
1	57072,19	57072,19	48	57080,11	57080,11
2	-	-	49	57090,24	57090,24
3	57075,15	57075,15	50	57077,18	57077,18
4	-	-	51	57077,13	57077,13
5	57072,24	57072,24	52	-	-
6	57077,09	57077,09	53	-	-
7	-	-	54	57095,17	57095,17
8	57075,10	57075,10	55	-	-
9	57075,22	57075,22	56	57075,11	57075,11
10	57077,11	57077,11	57	57095,20	57095,20
11	57075,20	57075,20	58	-	-
12	57072,14	57072,14	59	-	-
13	57077,15	57077,15	60	57095,20	57095,20
14	-	-	61	-	-
15	57077,11	57077,11	62	-	-
16	57077,09	57077,09	63	-	-
17	57077,11	57077,11	64	-	-
18	-	-	65	-	-
19	-	-	66	57095,15	57095,15
20	57077,18	57077,18	67	-	-
21	57077,12	57077,18	68	-	-
22	57077,12	57077,12	69	-	-
23	-	-	70	-	-
24	57077,08	57077,08	71	-	-
25	57077,09	57077,09	72	-	-
26	-	-	73	-	-
27	57077,17	57077,17	74	-	-
28	-	-	75	-	-
29	-	-	76	-	-
30	57072,21	57072,21	77	-	-
31	-	-	78	-	-
32	57075,21	57075,21	79	-	-
33	-	-	80	-	-
34	-	-	81	-	-
35	57078,20	57078,20	82	-	-
36	57080,11	57080,11	83	-	-
37	-	-	84	-	-
38	-	-	85	-	-
39	-	-	86	57090,26	57090,26
40	-	-	87	-	-
41	57075,08	57075,08	88	-	-
42	-	-	89	-	-
43	57090,22	57090,22	90	-	-
44	-	-	91	NaN	NaN
45	-	-	92	NaN	NaN
46	-	-	93	NaN	NaN
47	57075,20	57075,20			

L . Detección usando filtro unscented

Tabla 7.3: Resultados de épocas de detección en términos de MJD de las 93 supernovas del conjunto de 2015 de HiTS, usando el filtro unscented para una función $f(\Delta t) = \Delta t^{1.5}$ y $f(\Delta t) = \Delta t^2$.

Índ.	n=1.5	n=2.0	Índ.	n=1.5	n=2.0
1	-	57072,26	48	-	-
2	-	-	49	-	-
3	-	-	50	-	-
4	-	-	51	-	-
5	57072,24	-	52	-	-
6	-	-	53	-	-
7	-	-	54	-	-
8	-	57077,09	55	-	-
9	-	-	56	-	-
10	-	-	57	-	-
11	-	-	58	-	-
12	-	-	59	-	-
13	-	-	60	-	-
14	-	-	61	-	-
15	-	-	62	-	-
16	-	-	63	-	-
17	-	-	64	-	-
18	-	-	65	-	-
19	-	-	66	-	-
20	57090,24	57090,24	67	-	-
21	-	-	68	-	-
22	-	-	69	-	-
23	-	-	70	-	-
24	-	-	71	-	-
25	-	-	72	-	-
26	-	-	73	-	-
27	-	-	74	-	-
28	-	-	75	-	-
29	-	-	76	-	-
30	-	-	77	-	-
31	-	-	78	-	-
32	-	-	79	-	-
33	-	-	80	-	-
34	-	-	81	-	-
35	-	-	82	-	-
36	-	-	83	-	-
37	-	-	84	-	-
38	-	-	85	-	-
39	-	-	86	-	-
40	-	-	87	-	-
41	-	-	88	-	-
42	-	-	89	-	-
43	-	-	90	-	-
44	-	-	91	NaN	NaN
45	-	-	92	NaN	NaN
46	-	-	93	NaN	NaN
47	-	-			

M. Diagrama de clases de programa original

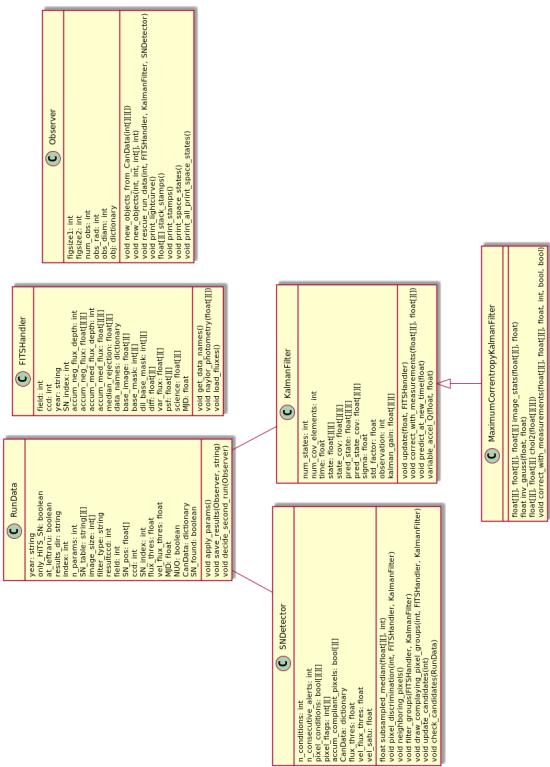


Figura 7.3: Diagrama de clases del programa original. Se listan cada uno de los métodos y variables de cada clase.

N . Diagrama de clases del programa refactorizado

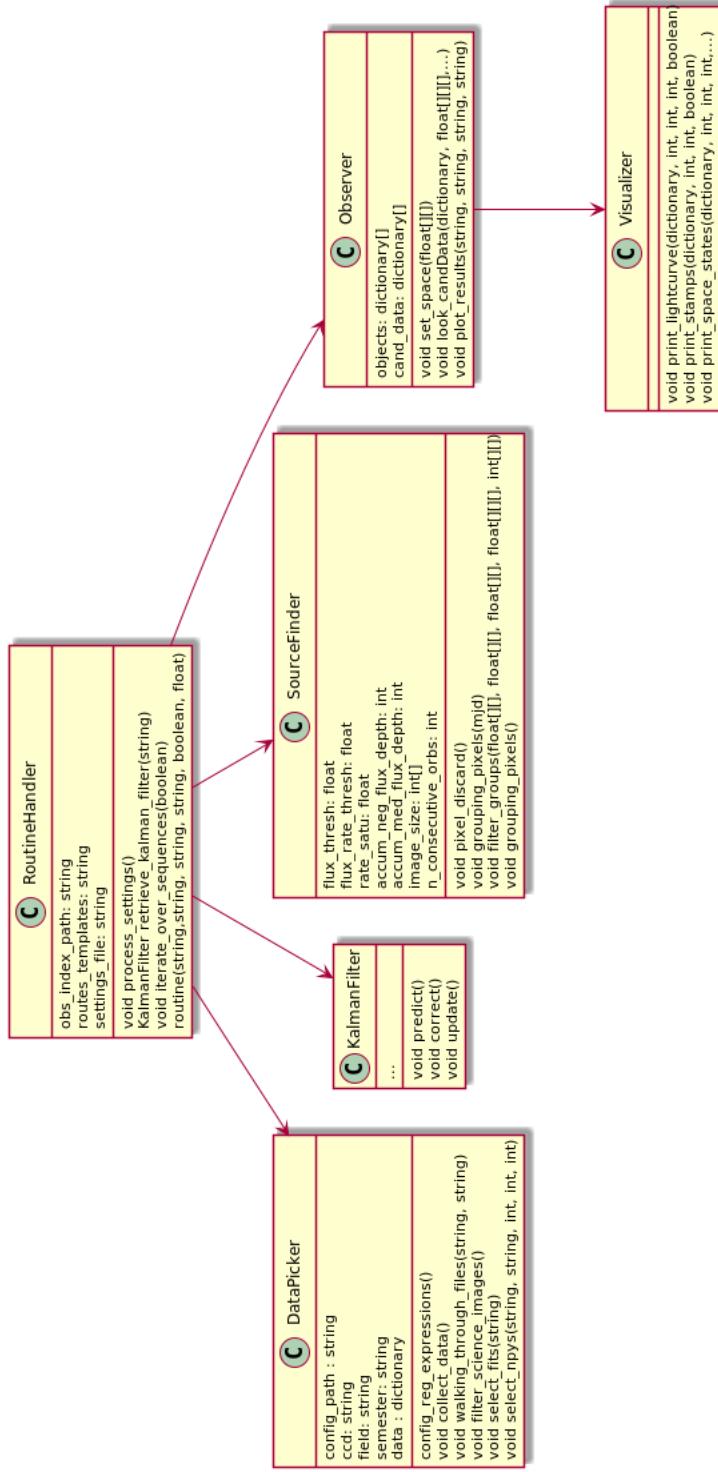


Figura 7.4: Diagrama de clases del programa refactorizado. Se muestra la relación entre las clases resultantes del proceso de refactoring. Se ha obviado la familia de filtros de Kalman por razones de dimensionalidad de imagen.