Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Grado en Ingeniería Informática

Diseño e implementación de ficheros de inicialización y configuración para las Cámaras y el control del Espejo Deformable de un Sistema de Óptica Adaptativa

Design and implementation of initialization and configuration files for Cameras and control of the Deformable Mirror of an Adaptive Optics System

Victoria Montserrat Manrique Rolo

La Laguna, 2 de Junio de 2021

D. **Jonay Tomás Toledo Carrillo**, con N.I.F. 78.698.554-Y profesor Titular de Universidad adscrito al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **Roberto López López**, con N.I.F. 36.026.296-Q Ingeniero Óptico adscrito al Departamento de Óptica en el Instituto de Astrofísica de Canarias, como cotutor.

D. **Esther Soria Hernández**, con N.I.F. 72.896.604-W estudiante de doctorado perteneciente al Departamento de Enseñanza en el Instituto de Astrofísica de Canarias, como cotutora.

C E R T I F I C A (N)

Que la presente memoria titulada:

"Diseño e implementación de ficheros de inicialización y configuración para las Cámaras y el control del Espejo Deformable de un Sistema de Óptica Adaptativa"

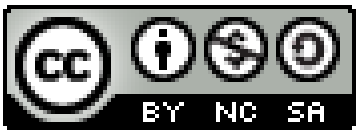
ha sido realizada bajo su dirección por D. **Victoria Montserrat Manrique Rolo**, con N.I.F. 43.385.261-Q.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 02 de 07 de 2021

Agradecimientos

Agradecer a Roberto López López y Esther Soria Hernández del Instituto de Astrofísica de Canarias por su colaboración durante el desarrollo de este proyecto, además agradecer a Jonay Toledo el tutor de prácticas y a Jesús Torres por su ayuda.

Licencia



Resumen

El objetivo de este proyecto es crear distintas aplicaciones para generar los ficheros de configuración tanto para la cámara de control como para el espejo deformable y la calibración del mismo de un Sistema de Óptica Adaptativa (SOA).

Para ello será necesario comprender el funcionamiento tanto del propio sistema como del software que lo controla.

Se utilizará la herramienta QtCreator para generar las interfaces gráficas y así facilitar la incorporación de este proyecto al software ya existente si fuera necesario, esta herramienta se puede utilizar tanto con el lenguaje de programación Python como C++, su elección dependerá de las necesidades del proyecto.

Como resultado final se deben haber creado dos aplicaciones, una para la creación de los ficheros tanto de la cámara como del espejo y otra para la calibración del espejo.

Palabras clave: óptica para adaptativa, polinomios Zernike, señales de control, pendientes, espejo deformable

Abstract

The goal of this project is to develop separate applications to generate configuration files for the control camera and the deformable mirror (DM) and the DM calibration for an Adaptive Optics System (SOA).

To achieve this goal, it will be necessary to understand the operation of both the system itself and the software that controls it.

The main development tool will be QtCreator that allows the creation of graphical interfaces and it will make future changes and updates easier as the already existing software was made with this tool as well. QtCreator can be used with Python or C++ as its main programming language and the choice will depend on the needs of the project.

As a final result there should be two applications, one that will generate the camera configuration files and another one that will handle the mirror calibration.

Keywords:optics, adaptative, Zernike polynomials, control signals, deltas, deformable mirror

Índice general

Índice de figuras

1. Introducción

En la astronomía existe un inconveniente, a la hora de poder observar la imagen de un objeto en el cielo ésta se ve distorsionada por las turbulencias de la atmósfera. Esto se puede resolver con una técnica llamada óptica adaptativa que mide y corrige las turbulencias atmosféricas en tiempo real haciendo uso de un espejo deformable que es manipulado por un conjunto de actuadores controlados por ordenador [1].

Además de corregir las distorsiones de la atmósfera, la óptica adaptativa es capaz de corregir los errores causados por la óptica del propio telescopio.

El Instituto de Astrofísica de Canarias está desarrollando un instrumento de Óptica Adaptativa que constituye el proyecto ALIOLI, que podrá incorporarse en telescopios de tamaño medio (1.5-2.5m). Sus componentes activos son una cámara de control, una de ciencia y un espejo deformable. El funcionamiento del sistema así como el de sus componentes se explicará con más detalle en el siguiente apartado. [2]

El objetivo de este proyecto es facilitar la inicialización del sistema, en concreto de la cámara de control y del espejo deformable, mediante el desarrollo de aplicaciones encargadas de generar los ficheros de configuración correspondientes a cada elemento y así evitar su modificación a mano y además desarrollar un sistema capaz de calibrar el espejo de manera automática.

1.1 Antecedentes y estado del arte

El Sistema de Óptica Adaptativa como ya se ha mencionado antes consta de tres componentes principales:

- Una cámara de control
- Un espejo deformable
- Una cámara de ciencia

El funcionamiento básico de este sistema es el siguiente. El frente de ondas procedente del objeto de observación es aberrado por la atmósfera. Éste es recogido a través de la apertura del telescopio. A continuación hacemos incidir este haz sobre un espejo deformable (DM), que permite modificar su superficie para contrarrestar la aberración generada por la atmósfera y por lo tanto poder obtener una imagen nítida del objeto a estudio. Para conocer la forma del haz incidente se utiliza un sensor de frente de onda (SFO).

Tenemos que tener calibrado el espejo deformable con el sensor para conocer la respuesta del sensor ante cambios en la superficie del DM y así poder enviar de manera continua al espejo la señal que producirá una geometría en su superficie tal que

contrarreste las aberraciones producidas por la atmósfera. Una vez este proceso esté terminado la imagen libre de aberraciones se podrá capturar mediante la cámara de ciencia.

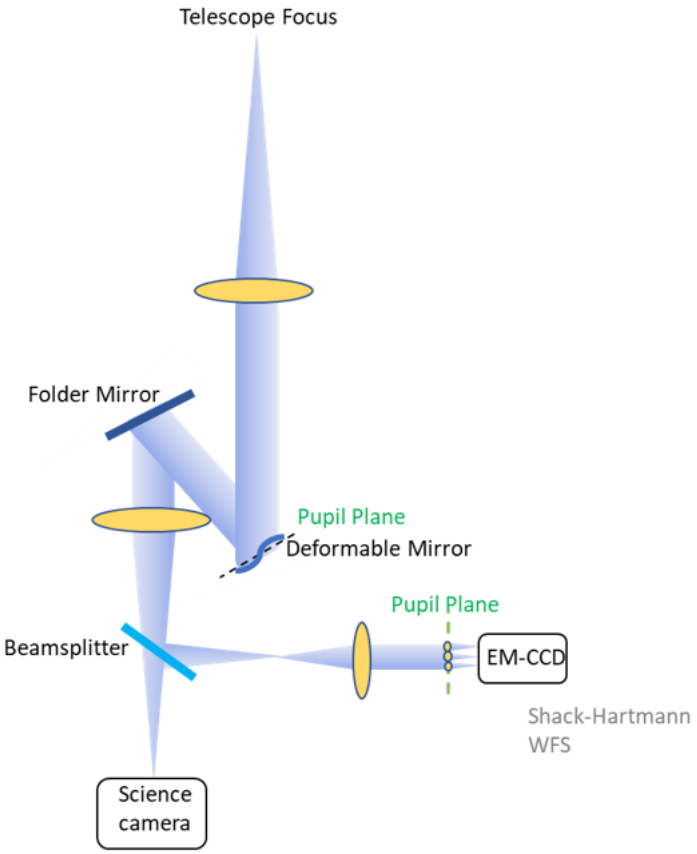


Figura 1. Esquema del Sistema de Óptica Adaptativa. Se muestra el espejo deformable (DM), la cámara de ciencia (Science camera) y la cámara de control (Shack-Hartmann SFO) [2]

A todo este sistema está asociada una aplicación.

1.1.1 FastCam+

FastCam+ es un software que controla el instrumento FastCam desarrollado por el Instituto de Astrofísica de Canarias y la Universidad Politécnica de Cartagena con el objetivo de obtener imágenes de alta resolución espacial dentro del espectro visible para telescopios terrestres.[3]

FastCam+ trabaja con detectores de la compañía Andor Technologies y tiene tres componentes.

- Andor Server: interfaz que trabaja directamente con la cámara y permite el control en remoto del proceso de adquisición. Esta se considera la interfaz principal.
- Andor Client: se conecta con la interfaz principal para poder controlar la cámara en remoto.
- FastCam: interfaz para post-procesado dedicada al método de Lucky Imaging y la aplicación de Shift and Add para la corrección de imágenes.

1.2 Objetivos Generales

Este proyecto consta de dos objetivos principales.

En primer lugar se desarrollará una aplicación que permite generar, cargar y modificar ficheros de configuración. Esta aplicación tendrá una versión en terminal y otra con interfaz gráfica y en segundo lugar para calibrar el sistema del espejo deformable se desarrollará una librería y a su vez una aplicación también con interfaz gráfica que se encargará de realizar los cálculos y correcciones necesarias.

Ambas aplicaciones se realizarán en Qt Creator para facilitar su posible modificación en el futuro debido a que los involucrados en el proyecto ya han trabajado con esta herramienta.

1.2.1 Herramientas utilizadas

Desarrollo

- Python v3.9.2
- Qt Creator v6.1 con Microsoft Visual C++ 2019
- C++17

2. Ficheros de configuración

En este capítulo se describe el proceso de desarrollo de la aplicación encargada de generar los ficheros de configuración de la cámara de control ANDOR.

2.1 Antecedentes

Las cámaras de las que dispone el Sistema de Óptica Adaptativa (SOA) son del fabricante ANDOR y de tipo Back-illuminated EMCCD. [4]

Esta cámara requiere de un fichero de configuración en el cual se determinan los parámetros con los que se debe trabajar. Este fichero consta de unas 500 variables aproximadamente que se pueden clasificar en cuatro categorías:

- Variables numéricas: dependiendo del parámetro se aceptarán números flotantes o números negativos.
- Variables de ruta: estas variables indican la ubicación de los directorios usados para guardar las imágenes capturadas y además la localización del propio programa que controla la cámara
- Variables de modo: aunque podrían formar parte de las variables numéricas este tipo de variables sólo acepta números enteros positivos dentro de un rango, cada número representa un modo de funcionamiento distinto de ciertas características.

2.2 Requisitos

Los encargados del proyecto ALIOLI determinaron los requisitos básicos para esta aplicación.

Se hizo una reunión con ellos a principios de abril en la cual se puso en contexto el proyecto y también se establecieron los requisitos necesarios.

Requisito nº1

Para que la aplicación pueda ser fácil de utilizar y apta para cualquier usuario de cualquier campo, se pide una interfaz gráfica. En ella se indica el nombre del parámetro y su valor el cual se podrá modificar.

Requisito nº2

La aplicación debe ser capaz de cargar y modificar ficheros ya existentes y crear ficheros nuevos usando una plantilla, el fichero generado además deberá tener una breve descripción de cada parámetro.

Requisito nº3

El fichero de configuración generado debe de poder ser editado a mano si fuera necesario.

Requisito nº4 (opcional)

La aplicación aparte de tener una versión con interfaz gráfica tendrá una versión que trabaje con la línea de comandos.

2.3 Desarrollo

Inicialmente se empezó a desarrollar la aplicación en C++ haciendo uso de la programación orientada a objetos para que las clases desarrolladas se pudieran usar tanto para el programa en terminal como para la aplicación generada con Qt Creator.

El fichero de configuración global consta en unas 565 líneas, cada una corresponde a un parámetro o a un título que categoriza los siguientes parámetros.

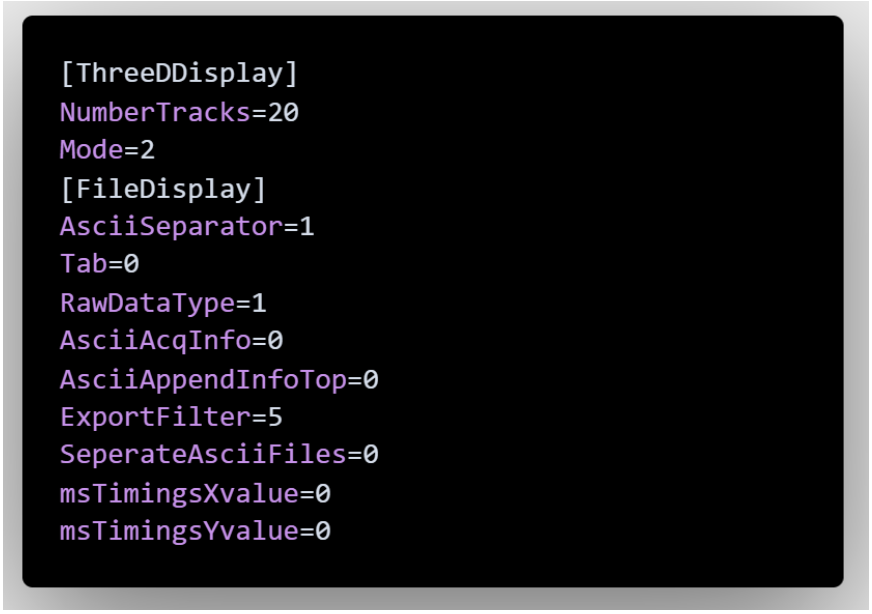


Figura 2. Fichero de configuración de la cámara Andor.

De todos los parámetros se suelen usar sólo 15 con frecuencia.

- Temperature Control
- Current Temperature
- Target Temperature
- Exposure Time
- Number of Accumulations
- Pre Amp Gain
- Frame Transfer Acquisition Mode
- Frame Gain
- Acquisition Mode
- Gain
- Baseline Camp
- EMGain Active
- Vertical Bin, Vertical End y Vertical Start
- Horizontal Bin, Horizontal y Horizontal Start

Como sólo se necesitan estas 15 variables, trabajar con el fichero completo resultaba innecesario.

Para facilitar el manejo de este fichero y además poder agregar información adicional de los parámetros, como su descripción y sus posibles valores aparte de su propio valor, se propuso a los miembros del proyecto realizar un fichero de configuración parcial que sirva como base para el fichero de configuración global, lo cual fue aprobado. El fichero parcial se realizó en formato JSON y como beneficio añadido, este cambio a formato JSON hizo que la manipulación del mismo fuera más fácil al trabajar con Python.

El fichero JSON contiene por cada variable, una breve descripción, los valores posibles aceptados y el valor real.

Ej:

```
"AcquisitionMode": {
```

```
    "description": "Modo de adquisición",
    "inputs": "Números enteros desde el 0 hasta el x",
    "value": 3
}
```

La idea de este fichero parcial es que el usuario pueda usar este fichero para cambiar los parámetros importantes y a partir de éste, generar el fichero de configuración global, que es leído por la cámara, con la aplicación.

2.4 Primer prototipo

Se realizó un primer prototipo en terminal usando Python 3.9, se trata de una aplicación muy básica que muestra los parámetros con un valor cargado de un fichero por defecto que contiene los valores más usuales y la opción de modificarlo. Este constituye la función principal.

El programa muestra una lista con los parámetros y un número el cual será el identificador que permitirá cambiar el valor del parámetro.

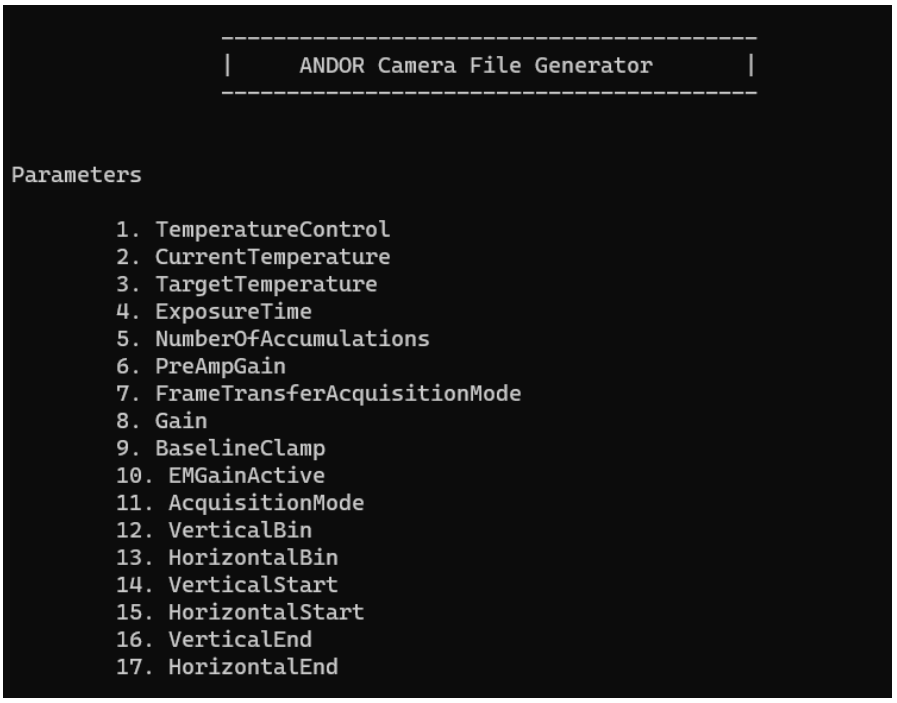


Figura 3. Lista de parámetros con su identificador correspondiente.

El menú de opciones de la aplicación es el siguiente:.

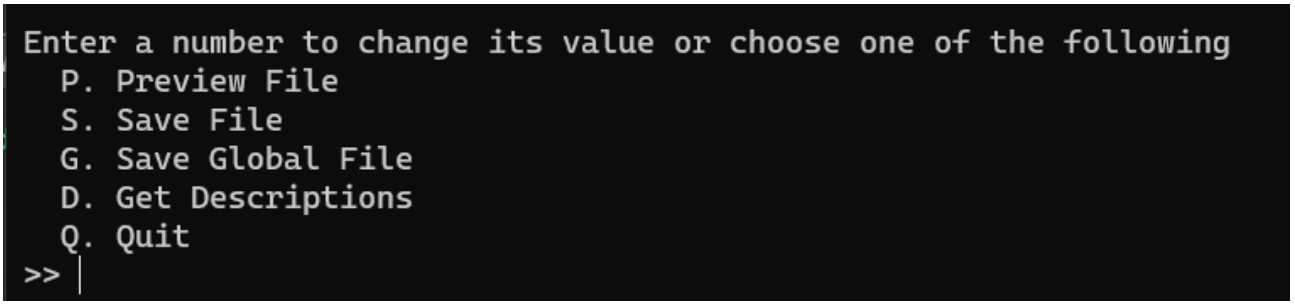


Figura 4. Menú de opciones

Como se puede ver la Figura 4 para cambiar el valor de un parámetro simplemente se tendrá que introducir el identificador que le corresponde al parámetro según la lista mostrada en la Figura 5.

En caso contrario se podrá elegir alguna de las opciones que permite previsualizar el fichero parcial con sus valores, generar el fichero de configuración tanto parcial como global, mostrar las descripciones de cada parámetro, se podrán mostrar todos los parámetros o uno en específico y finalmente salir del programa.

Este primer prototipo fue presentado en una reunión de seguimiento a mediados de abril y fue aprobado por el equipo del proyecto y esto concluía el código base que se encargaba de manipular los ficheros y el siguiente paso sería la interfaz gráfica.

2.5 Segundo prototipo

Uno de los requisitos establecidos fue que la interfaz gráfica fuera fácil de usar e intuitiva, para lograr esto se realizó una interfaz la cual muestra el nombre de cada parámetro y un campo de texto que permite introducir el valor deseado.

La aplicación carga unos valores guardados en un fichero de configuración por defecto que son los más usuales.

Además, como en la aplicación de terminal, se permite cargar un fichero parcial para su modificación y generar un fichero de configuración tanto parcial como global.

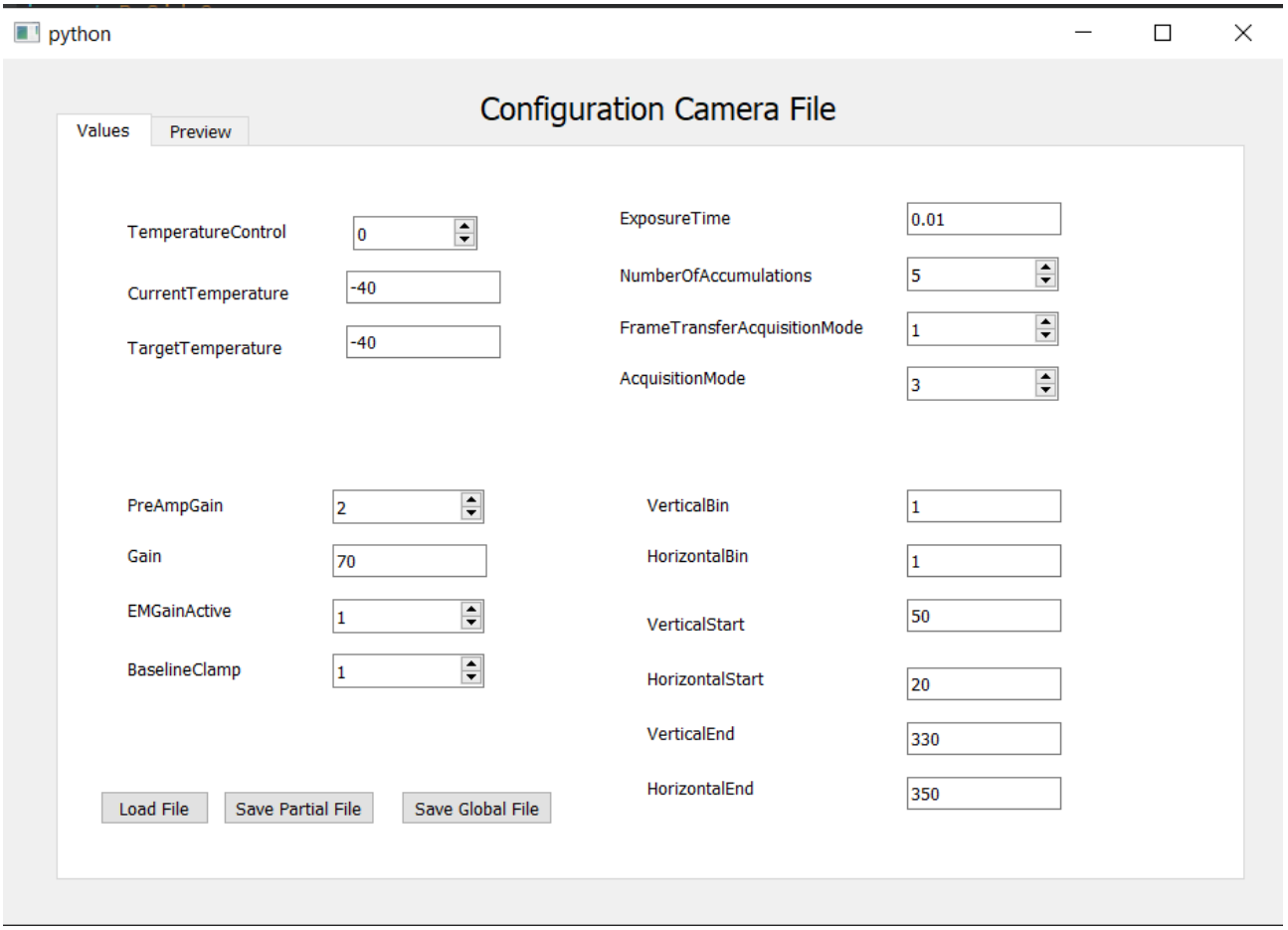


Figura 5. Interfaz gráfica de la aplicación de generación de ficheros.

Después de aprobar el segundo prototipo se probó su funcionalidad. Se comprobó que la cámara leía bien el fichero generado por la aplicación y que todos los ficheros parciales se generaban bien y se dio por concluida esta parte del trabajo.

3. Calibración del espejo deformable

En este apartado se explicará el desarrollo del software de calibración del espejo deformable.

Antes de dar paso al desarrollo del código se explicará qué es un espejo deformable y su funcionamiento. Después se explicará el proceso por el cual se consigue la calibración y finalmente se presentará el código desarrollado con interfaz integrada.

3.1 ¿Qué es un espejo deformable?

Un espejo deformable es un espejo cuya superficie puede ser deformada para modificar la geometría del haz incidente. Los espejos deformables se utilizan en combinación con sensores de frente de onda y sistemas de control en tiempo real para su aplicación en un

sistema de óptica adaptativa.

En la práctica la forma del espejo deformable debería cambiar mucho más rápido que el proceso de corrección ya que este proceso puede llevar varias iteraciones.

La superficie de un espejo deformable se modifica mediante los actuadores mencionados anteriormente, cada actuador se mueve independientemente. Dependiendo del tipo del espejo deformable los actuadores pueden tener movimientos distintos.[5]

Si se trata de un espejo deformable segmentado significa que cada actuador tiene dos ejes que se mueven hacia arriba o abajo modificando su ángulo con respecto a la base del espejo, también estos actuadores no influyen de ninguna forma a sus vecinos ya que cada actuador tiene su propia superficie cada movimiento afecta a un único actuador.

Otro tipo de espejo deformable sería el continuo, este espejo deformable tiene como característica que todos los actuadores se encuentran en una lámina reflectante. Los actuadores sólo tienen un eje que les permite moverse hacia arriba o hacia abajo y al encontrarse en una lámina cada movimiento de un actuador afecta a sus vecinos.

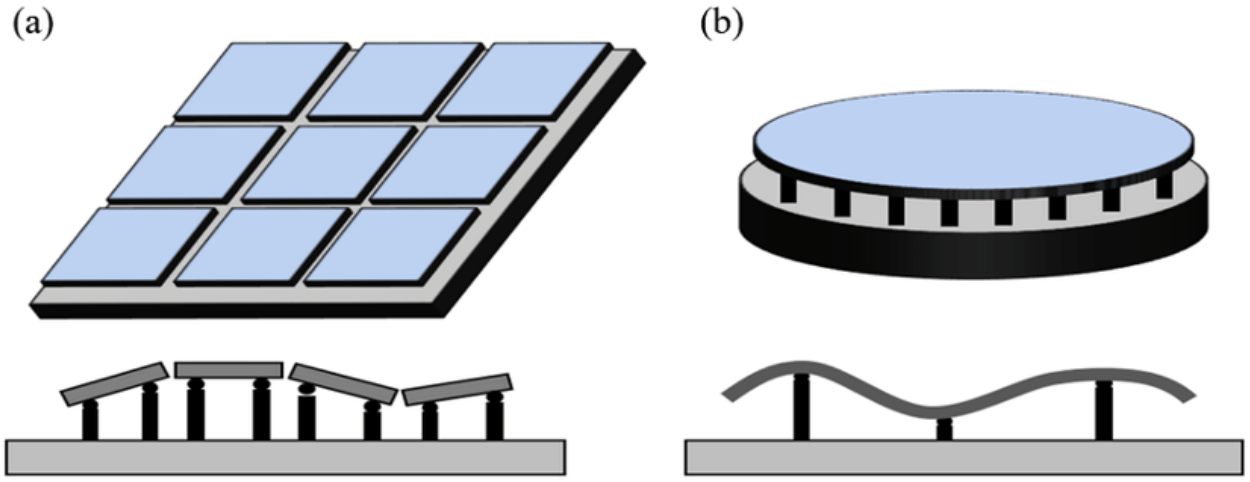


Figura 6. Disposición de los actuadores de un espejo deformable segmentado (a) y continuo (b). [6]

Los parámetros que incluye un espejo deformable son:

- Número de actuadores: determina el número de grados de libertad que el espejo puede corregir.
- Actuator pitch: es la distancia que hay entre los centros de los actuadores.
- Actuator stroke: es el desplazamiento máximo de cada actuador, generalmente se representa como positivo o negativo desde su posición central.
- Función de influencia: es la forma característica que corresponde a la respuesta del espejo a la acción de un solo actuador.
- Actuator coupling: muestra cómo afecta el movimiento de un actuador sobre sus vecinos.

- Tiempo de respuesta: mide cuánto tarda el espejo en reaccionar a la señal de control.

El espejo que está incluido en el Sistema de Óptica Adaptativa del IAC se trata de un un modelo continuo de la casa Alpao, con 88 actuadores.

Ahora que ya sabemos que es un espejo deformable explicaremos el proceso de calibración.

3.2 Calibración

3.2.1. Método de calibración

El proceso de calibración y control del espejo deformable se puede realizar con un circuito abierto o cerrado y su principal diferencia es la disposición de los componentes entre sí.

En un sistema de circuito cerrado, el cual es usado en el sistema en el que se está trabajando, la luz que regresa de la muestra pasa por el espejo deformable antes de pasar por el sensor, esto significa que el sensor mide el error en la corrección de la aberración en particular, en vez de la aberración total y hace que el sensor proporcione información sobre si el espejo ha alcanzado la forma deseada para obtener un frente de onda plano. En un sistema de circuito abierto, no hay retroalimentación sobre si el espejo tiene la forma correcta o no ya que la luz no pasa por el espejo al volver de la muestra.

La ventaja de utilizar el sistema de circuito cerrado es que puede tener más precisión y cualquier aberración se compensa por sí sola.

El sistema con el que se está trabajando implementa el método de circuito cerrado.

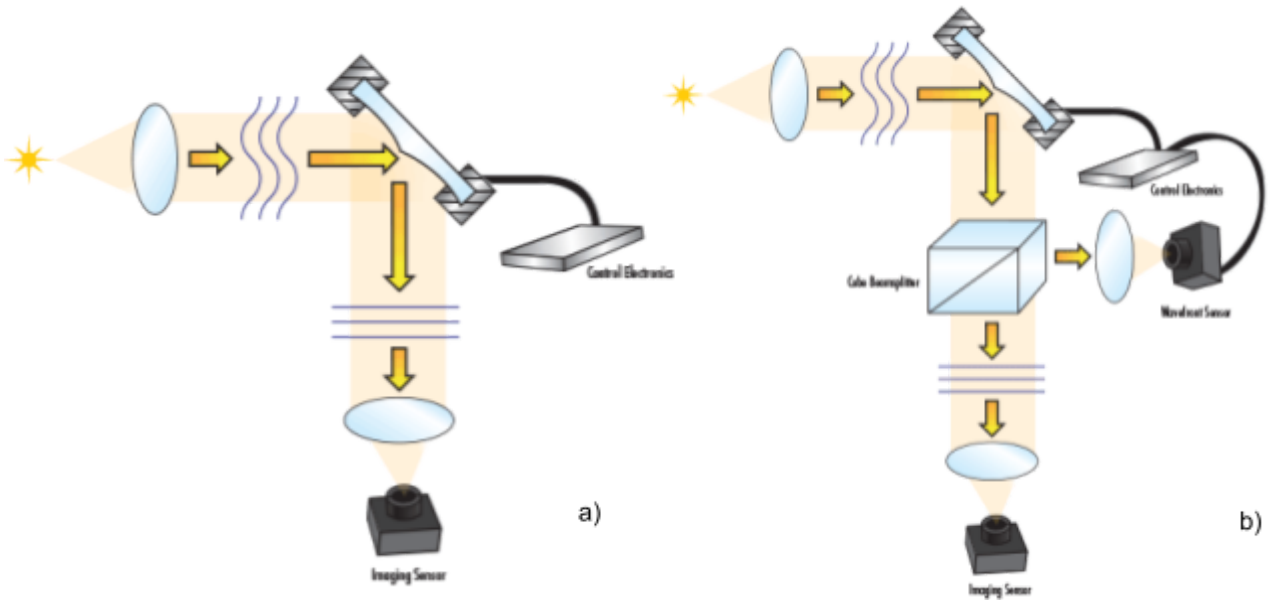


Figura 7. Diferencia entre un sistema de circuito abierto (a) y un sistema de circuito cerrado (b) [7]

3.2.2. Proceso

El objetivo del procedimiento de calibración es determinar la relación entre la señal de control enviada al espejo deformable y la respuesta que produce sobre el sensor, y por lo tanto sobre la imagen de ciencia. Si conocemos la respuesta del espejo podremos calcular la señal que hay que enviarle para corregir un frente de ondas aberrado.

Matriz de funciones de influencia

Para este proyecto el tipo de caracterización que vamos a hacer es zonal. Consiste en ir moviendo cada actuador de manera independiente y leer en el sensor de frente de onda (SFO) la respuesta al cambio inducido. Al aplicar una señal de control a un actuador dado, la superficie alcanza una forma que se denomina función de influencia. La suma de todas las funciones se puede considerar como la superficie espejo y esto hace que la medición de las funciones de influencia caracterice las propiedades del espejo.

Antes de medir estas funciones de influencia el espejo se coloca en su posición de espejo plano, que suele coincidir con el valor cero ya que el actuador es impulsado usando valores positivos y negativos, para así poder realizar movimientos hacia delante o hacia atrás sin llegar a su desplazamiento máximo o mínimo durante el proceso de calibración y todos los movimientos se harán con relación a la posición media.

La función de influencia para cada actuador forma una columna de la matriz de funciones de influencia y se puede expresar en términos de pendientes (IFS) o de valores de los coeficientes de Zernike (IFZ) que reproducen la geometría de frente de ondas. Esto dependerá de la configuración de salida del módulo de sensado.

Cada elemento de la matriz IFS está representado por:

$$IFS_{l,Actuator} = \frac{S_{lx}^{+C} - S_{lx}^{-C}}{2C} \quad (1)$$

donde l es el número de microlentes, $S_{lx}{}^{+C}$ y $S_{lx}{}^{-C}$ son las pendientes en la dirección x para la microlente l cuando la señal de control positiva o negativa C se aplica con relación a la posición de mitad de carrera.

A su vez los elementos de la matriz IFZ se representan de la siguiente manera:

$$IFZ_{coeff,Actuator} = \frac{a_{coeff}^{+C} - a_{coeff}^{-C}}{2C} \quad (2)$$

donde a_{coeff}^{+C} y a_{coeff}^{-C} son los coeficientes de Zernike para el coeficiente número $coeff$ número cuando se aplica una señal de control C positiva o negativa en relación con la señal de control de mitad de carrera.

Esta matriz tendrá tantas filas como número de pendientes o en su defecto número de polinomios de Zernike se hayan obtenido y tantas columnas como número de actuadores tenga el espejo deformable.

Matriz de control

Una vez calculada la matriz de funciones de influencia se obtendrá una matriz de control que contiene las señales de control C, se puede generar con la matriz de influencia generada a partir de las pendientes o a partir de los coeficientes de Zernike.

$$C = (IFS^{+})S \text{ (3)} \quad C = (IFZ^{+})a \text{ (4)}$$

Donde IFS^{+} es la pseudo-inversa de la matriz de funciones de influencia en términos de las pendientes y S es un vector de pendientes medidas y de la misma manera IFZ^{+} es la pseudo-inversa de la matriz de funciones de influencia en términos de los coeficientes de Zernike y a son los polinomios de Zernikes, esta matriz se considera la matriz de control que tendrá tantas filas como número de actuadores y tantas columnas como pendientes o coeficientes de Zernike.

Cuando realicemos una medida de un frente de ondas aberrado que incide sobre nuestro sistema, para obtener los valores de cada actuador se deberá multiplicar la matriz de control por las pendientes o por los polinomios de Zernike resultando en un vector columna con tantas filas como actuadores donde se indica la amplitud de cada uno de ellos. Cuando realizamos este proceso de forma continua hablamos de que hemos cerrado el lazo del sistema. [8]

Todas estas matrices se relacionan de la siguiente manera:

$$ZA = F \text{ (5)}$$

Donde la Z representa la matriz de influencia, A el vector con los voltajes de los actuadores y F es la lectura del WFS. Como el objetivo de todo este proceso es lograr la superficie idónea del espejo el objetivo principal es obtener los voltajes. Según la expresión (5) la manera más fácil de obtener estos valores es despejando en la ecuación. Siguiendo las reglas matemáticas para despejar A nos sería necesario realizar:

$$Z^{T}ZA = Z^{T}F \text{ (6)}$$

$$A = (Z^{T}Z)^{-1}F \text{(7)}$$

Donde $(Z^TZ)^{-1}$ representa la pseudo-inversa, es por esto que es necesario calcular esta matriz para poder lograr la calibración del DM. [8]

3.3 Implementación

3.3.1. Datos y librerías necesarias

Como se ha explicado anteriormente hay dos maneras de obtener las matrices necesarias para calibrar el espejo, mediante las pendientes o mediante los polinomios de Zernike. Después de una reunión con los responsables del SOA, se llegó a la conclusión de que usar los polinomios de Zernike sería más conveniente y además más sencillo ya que los polinomios trabajan con una dimensión y las pendientes tienen un valor x y un valor y.

Para obtener los polinomios de Zernike es necesario tener acceso a la cámara y a su vez generar dichos polinomios. Esto lo logramos mediante dos librerías:

- **AOCamera:** establece una conexión con la cámara de control, permite capturar imágenes así como cambiar parámetros de captura como la exposición y la ganancia.
- **AOProcessing:** Procesa datos generados a partir de una imagen capturada y esto incluye los deltas (pendientes) y los polinomios de Zernike.

También será necesario tener acceso al propio espejo, la librería permite realizar varias operaciones básicas sobre el espejo. Permite inicializar el espejo, hacer reset, es decir asignar a todos los actuadores su posición por defecto, que suele ser cero, cambiar los valores de los actuadores y devolver propiedades del mismo como el número de actuadores, su amplitud etc. Cabe destacar que todas las operaciones que se realizan sobre los actuadores son de escritura, no existe ninguna de lectura.

Una interfaz llamada *DM_BOL* utiliza la librería nativa del espejo y facilita todas las operaciones anteriormente mencionadas.

3.3.2. Generación de matrices y de valores de actuadores

Matriz de influencia

Antes de generar la matriz de influencia es necesario definir unos parámetros:

- num_push_pull: número de veces que un actuador se mueve negativamente y positivamente, por cada unidad se realiza ambos movimientos.
- num_measure: número de medidas tomadas desde la cámara, esto devuelve los polinomios de Zernike o los deltas de la imagen actual.
- amplitude: amplitud total del actuador
- num_zernikes: número de polinomios de Zernike, se usa para definir el tamaño de las matrices.

El algoritmo consiste en, por cada actuador se realizan tantos push/pull como se indica, los movimientos representan las señales de control del actuador que son de la misma magnitud pero de distinto signo dentro del intervalo [-1,1].

Por cada movimiento se realizarán las medidas indicadas por *num_measures* , donde se moverá el actuador con la señal de control correspondiente y se obtendrá una medida. Al haber la posibilidad de hacer varias mediciones estas se irán añadiendo en una matriz y al salir de este proceso se realizará una media de todas las medidas acumuladas.

Una vez terminada esta parte se normalizan todas las lecturas y se aplica la fórmula mostrada en (2). Al terminar el movimiento push/pull se acumulan las medidas en un vector temporal y después se añade al actuador correspondiente.

```
fun influence_matrix:
    for i in num_actuators:
        for push_pull in num_push_pull:
            for move in [-1,1]:
                lec t= matrix(num_zernikes,num_measures)
                lec=np.zeros(num_zernikes)
                for medida in range(num_measures):
                    deformable_mirror.actuator[i] = move * amp #movemos el espejo
                    lect.col(medida)= camera.take_reading() #tomamos medida
                lec=mean(lect)
                influ_M_temp += s * lec / (2 * amp) #corresponde a la formula
            medida.col(h)=influ_M_temp //
        influ_M.col(ind)= sum(medida)/num_push_pull
    return influ_M
```

Figura 8. Pseudocódigo para la generación de la matriz de influencia.

Matriz de control

Para el manejo de matrices se utilizó la librería *Eigen* se trata de una librería para álgebra lineal que trabaja con matrices y vectores además, implementa algoritmos relacionados [9].

Esta librería nos permite hacer operaciones aritméticas entre matrices así como realizar reducciones (como la media o suma por filas/columnas de una matriz) y lo más importante nos permite realizar la pseudo-inversa que corresponderá a la matriz de control.

Para realizar la pseudo-inversa se pueden usar dos métodos.

Descomposición en valores singulares (SVD)

La descomposición en valores singulares se trata de una factorización de una matriz, se dice que cualquier matriz A de m filas y n columnas se puede factorizar en:

$$A = USV^{T} \quad (4)$$

Gracias a esta factorización y a la técnica de mínimo cuadrados se puede hallar la pseudo-inversa de una manera fácil siguiendo la siguiente fórmula. [10]

$$A^{+} = VS^{+}U \quad (5)$$

La librería *Eigen* es capaz de realizar la descomposición mediante la llamada al método *jacobiSvd* al cual le indicamos que compute las matrices U y V.

Una vez hecha la descomposición sólo será cuestión de aplicar la fórmula pero se tendrá que incluir una tolerancia ya que la matrices contienen valores muy pequeños y es probable que los cálculos internos resulten en cero.

```
double threshold = 0.0001;

auto svd = influ_mat_.jacobiSvd(Eigen::ComputeThinU | Eigen::ComputeThinV);
double min = threshold * (std::max(mat_eigen_.cols(), mat_eigen_.rows())) * svd.singularValues().array().abs()(0);
control_mat_ = svd.matrixV() *
        (svd.singularValues().array().abs() > min).select(svd.singularValues().array().inverse(),0).matrix().asDiagonal() *
        svd.matrixU().adjoint();
```

Figura 9. Implementación de obtención de la pseudo-inversa mediante la descomposición en valores singulares en C++.

Implementación de Eigen x

La librería Eigen incluye un método que permite realizar la pseudo-inversa la cual requiere una descomposición ortogonal completa de la matriz antes de hacer la pseudo-inversa.

```
Eigen::MatrixXd A = data;
Eigen::MatrixXd pinv = A.completeOrthogonalDecomposition().pseudoInverse();
```

Figura 10. Método para obtener la pseudo-inversa con las funciones nativas de Eigen.

Valores de actuadores

La generación de los valores de actuadores es muy sencilla, sólamente se debe multiplicar la matriz de control por una lectura realizada, (pendientes o Zernikes dependiendo de cómo hayamos generado las matrices) y esto generará un vector con

tantas posiciones como actuadores.

3.3.3. Simulaciones y pruebas

Antes de probar la generación de matrices con datos reales se hicieron varias simulaciones con una lectura de la cámara guardada en fichero.

Primera simulación

La primera simulación tenía como objetivo comprobar si el proceso de generar las matrices de influencia, la de control y los valores de los actuadores se hacía correctamente sin errores de compilación y la manipulación y operaciones con matrices eran correctas. Para ello se hizo un fichero con medidas tomadas de la cámara, estas medidas son simuladas por un programa de python, con los siguientes parámetros.

La lectura de la cámara fue obtenida con los siguientes valores

- num_push_pull: 2
- num_measure: 1
- amplitude: 0.1
- movimientos: [-0.1, 0.1]
- número de Zernikes: 15
- número de actuadores: 81

Esto se guardó en una matriz que por cada actuador tenía cuatro lecturas y tenía un tamaño de 15x324. Se generó la matriz de influencia, obviando el proceso de mover los actuadores del espejo y el de tomar las medidas de la cámara ya que vienen dadas por el fichero y dió como resultado una matriz de 15x81, al hacerle la inversa, generó una matriz de 81x15 y el vector de los valores de los actuadores tenía un tamaño de 81. Esto significaba que el código manipulaba bien las matrices.

Segunda simulación

En la segunda simulación el objetivo era comprobar si la matriz de control, es decir la pseudo-inversa, se generaba correctamente, usando los distintos métodos disponibles y si producía valores de actuadores correctos.

Esto se comprobó de la siguiente manera, primero realizamos la calibración del espejo deformable por lo que obtenemos su matriz de influencia.

A continuación, partiendo de un frente de ondas ideal, se modificó su geometría enviando una señal aleatoria a los actuadores del DM que llamamos *h_actuators*. En esta simulación el único elemento que contribuye a la geometría del frente de ondas es el DM. Por último procedimos a la lectura del frente de ondas en el WFS.

El objetivo era generar la matriz de control y las posiciones de los actuadores que reproducen la geometría del frente de onda incidente. A la hora de graficar la superficie del espejo el gráfico generado por *h_actuators* y por la lectura del frente de onda debería ser lo más similar posible. Se realizaron pruebas generando la pseudo-inversa mediante el método implementado en *Eigen y* el método SVD siendo este último el que dio mejores resultados.

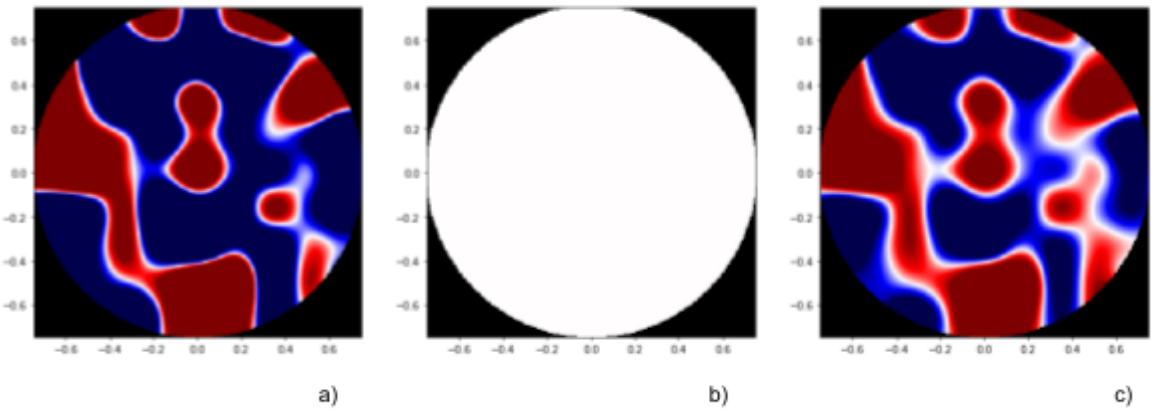


Figura 11. Comparación de las superficies del espejo.

Se muestran varias superficies del espejo: una aleatoria a) que va a ser imitada, una plana b) y la superficie correspondiente a la de los valores generados c).

Como se puede ver las superficies son prácticamente iguales por lo que esta simulación fue realizada con éxito.

3.3.4. Pruebas con el espejo

Para comprobar que la conexión del espejo se realizaba correctamente y corregir posibles fallos de implementación con la interfaz *DM_BOL* se realizaron las siguientes pruebas que consisten en operaciones básicas:

- Inicialización del espejo
- Reset del espejo
- Mover actuadores

Adicionalmente se hizo un barrido que consiste en recorrer todos los actuadores y por cada uno moverlo a su posición máxima, mínima y dejarlo en su posición cero.

Estas pruebas no dieron fallo y con este paso se dió por concluida la fase de pruebas

3.4 Aplicación

Una vez realizadas todas las pruebas y correcciones necesarias se empezó el desarrollo de la aplicación con interfaz gráfica. Se establecieron los siguientes requisitos.

Requisito nº1
La aplicación debe ser capaz de calibrar el espejo deformable en un tiempo mínimo de manera automática.

Requisito nº2
Se podrá cambiar las señales de control de un actuador específico si fuera necesario.

Requisito nº3
Se podrá realizar un barrido de los actuadores.

3.4.1. Interfaz Gráfica

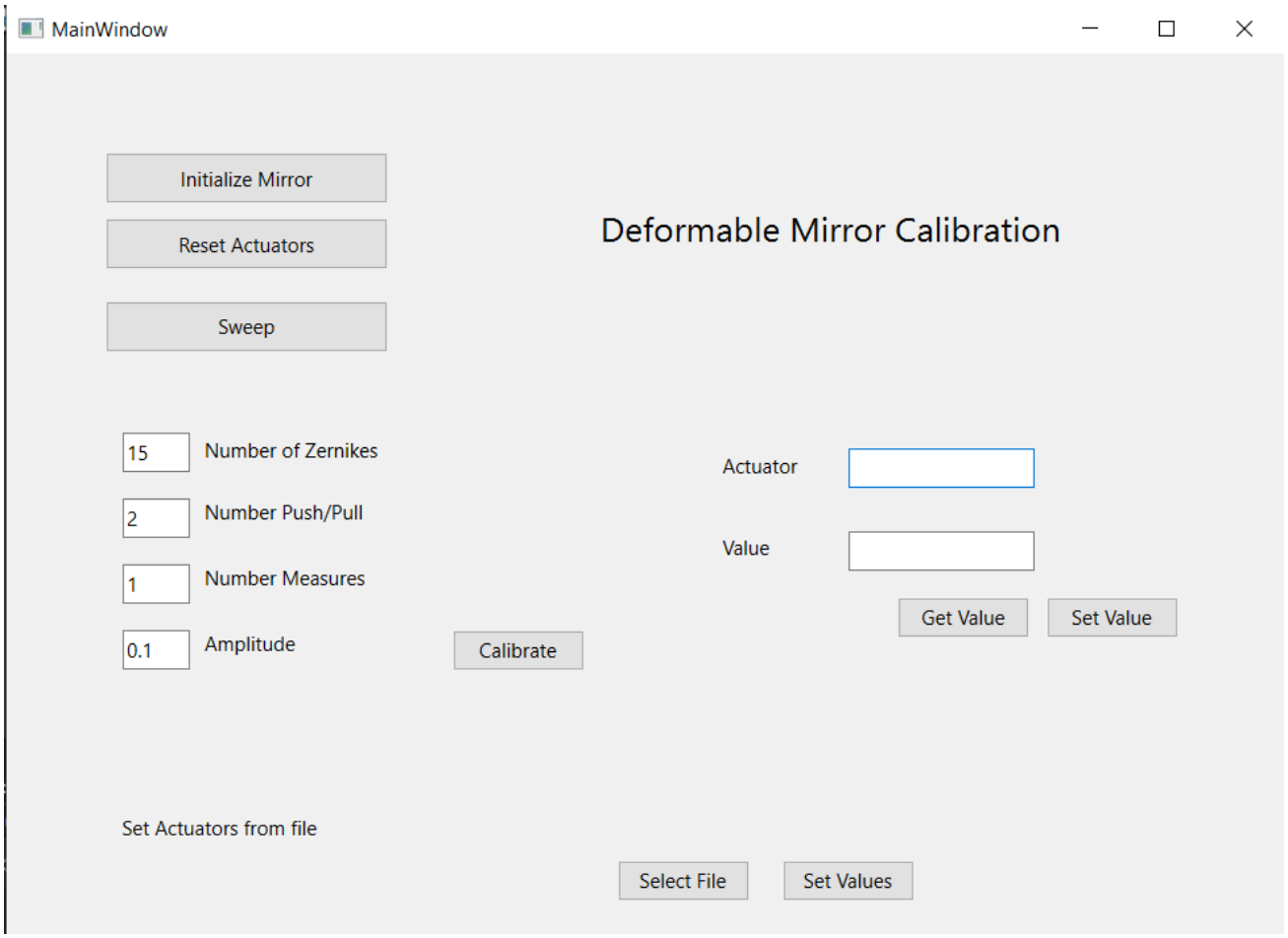


Figura 12. Interfaz gráfica de la aplicación de calibración del espejo deformable.

La aplicación permite inicializar el espejo (Initialize Mirror), reiniciar los actuadores (Reset Actuators) y hacer un barrido (Sweep).

También recuperar el valor de un actuador específico introduciendo el número correspondiente en el campo “Actuator” y al hacer click en “Get Value” el valor del mismo

aparecerá en el campo de texto de “Value”. De forma similar se permite asignar un valor a un actuador específico introduciendo el número de actuador y su valor y haciendo click en “Set Value”.

Para la calibración del espejo se muestran los diferentes parámetros que se necesitan para generar las matrices y sólo será necesario hacer click en “Calibrate”.

Adicionalmente se podrá asignar el voltaje de los actuadores desde un fichero.

3.4.2. Workflow

Una ejecución típica de la aplicación sería la siguiente:

En primer lugar se debe inicializar el espejo deformable para evitar errores de conexión, el siguiente paso será hacer un reset de los actuadores y así dejar el espejo “plano” y hacer un barrido para comprobar que todos los actuadores están funcionando correctamente.

Una vez realizado el proceso inicial de configuración el próximo paso será la calibración, se indicará el valor de los parámetros necesarios o se dejarán los valores que vienen por defecto.

Cuando el espejo esté calibrado, si existe algún actuador que esté fuera de lugar y se pueda mejorar manualmente se asignará el valor usando las funciones de “Get value” y “Set value”.

Para que la calibración del espejo sea realizada correctamente se deberá seguir estos pasos en el orden estipulado ya que sino no se seguiría el método de calibración de circuito cerrado y después de todo esto damos por concluida esta fase del proyecto.

4. Resultados finales

Al concluir la fase de desarrollo de cada aplicación se hizo una reunión con los responsables del proyecto para mostrar el resultado final y realizar las pruebas finales, se probaron las aplicaciones tanto con el instrumento como sin él, en el caso de la aplicación de calibración del espejo, y se probaron todas sus funcionalidades.

Estas pruebas resultaron exitosas, se declaró la fase de desarrollo como terminada, lo siguiente a realizar era la entrega del código con su documentación correspondiente y con esto queda concluido este trabajo.

Conclusiones y líneas futuras

El desarrollo de este trabajo ha supuesto un gran reto y una gran responsabilidad, desarrollar código para un sistema que va a ser utilizado por muchos usuarios en un futuro hace que los niveles de responsabilidad crezcan. Ha permitido profundizar el conocimiento sobre tecnologías y herramientas ya conocidas y además desarrollar código en base a unos requisitos propuestos por lo que se podría denominar un cliente ha sido una experiencia totalmente nueva.

Los objetivos generales se han cumplido, la primera aplicación es capaz de generar ficheros de configuración de la cámara de control y la segunda es capaz de calibrar el espejo deformable del sistema.

A pesar de haber cumplido los objetivos hay ciertos aspectos que se pueden mejorar como por ejemplo la generación de matrices, en concreto la pseudo-inversa, se podría generar con otro método como por ejemplo la regularización de Tikhonov que controla posibles problemas con valores cercanos a cero.

Otro aspecto que sería interesante sería realizar un estudio estimando el tiempo que tarda el programa en leer la imagen, calcular las medidas, realizar los cálculos de las matrices y enviar los voltajes a los actuadores y ver si el orden de magnitud es menor a los 30ms, lo que corresponde al tiempo en el que cambia la atmósfera.

Finalmente mencionar que este trabajo forma parte de un conjunto de TFGs asociados al proyecto ALIOLI, uno de ellos fue el desarrollo de librerías de acceso a la cámara de procesamiento datos del cual se beneficia este trabajo y otro trabajo que permite el graficado de distintos elementos, que es beneficiado por este trabajo y la unión de todos ellos constituyen el software que permite que el sistema funcione correctamente.

Summary and Conclusions

Developing code for a system that will be used by many users in the future increases the level of responsibility. It has allowed to deepen the knowledge about technologies and tools already known and also to develop code based on requirements proposed by what could be called a client has been a totally new experience.

The general objectives have been met, the first application is capable of generating configuration files for the control camera and the second is capable of calibrating the system's deformable mirror.

Despite having met the objectives, there are certain aspects that could be improved, such as the generation of matrices, specifically the pseudo-inverse, which could be generated with another method such as the Tikhonov regularisation that controls possible problems with values close to zero.

Another aspect that would be interesting would be to carry out a study estimating the time it takes for the program to read the image, calculate the measurements, perform the matrix calculations and send the voltages to the actuators and see if the order of magnitude is less than 30ms, which corresponds to the time in which the atmosphere changes.

Finally, it is worth mentioning that this work is part of a set of TFGs associated with the ALIOLI project, one of which was the development of libraries for accessing the camera and processing the data, from which this work benefits, and another work that allows the graphing of different elements, which benefits from this work, and the union of all of them constitutes the software that allows the system to function correctly.

Presupuesto

Configuración de la cámara

Tareas:

- Programa en terminal con funcionalidades, 15 horas de trabajo
- Interfaz gráfica en QtCreator, 10 horas de trabajo

Calibración de espejo deformable

Tareas:

- Código principal, 75 horas
    - Generación de matrices, 20 horas
    - Operaciones con el espejo, 15 horas
    - Realizar conexiones entre componentes, 30 horas
    - Testing, 10 horas
- Interfaz gráfica en QtCreator, 20 horas

Las horas indicadas representan una media de las horas dedicadas a cada tarea, en total al trabajo se han dedicado unas 120 horas aproximadamente con una valoración de 8€/hora.

Bibliografía

[1] C. de los proyectos Wikimedia, “Óptica adaptativa,” *Wikipedia*, Oct. 22, 2019. https://es.wikipedia.org/wiki/%C3%93ptica_adaptativa .

[2] E. Soria, R. L. López, A. Oscoz, and C. Colodro-Conde, “ALIOLI: presentation and first steps.” http://www.spiedigitallibrary.org/conference-proceedings-of-spie/11448/114482E/ALIOLI-presentation-and-first-steps/10.1117/12.2562277.short .

[3] A. Oscoz *et al.*, “FastCam: a new lucky imaging instrument for medium-sized telescopes.” http://www.spiedigitallibrary.org/conference-proceedings-of-spie/7014/701447/FastCam--a-new-lucky-imaging-instrument-for-medium-sized/10.1117/12.788834.short .

[4] “iXon EMCCD Cameras,” *Oxford Instruments*. https://andor.oxinst.com/products/ixon-emccd-cameras?gclid=Cj0KCQjw5uWGBhCTARIsAL70sLLGyHg9DlYDPC_h5lYmToNyXNhp8yQ4lzMfjpnEw1srWtsYDLnbTmwaAoC6EALw_wcB .

[5] Contributors to Wikimedia projects, “Deformable mirror,” *Wikipedia*, May 16, 2021. https://en.wikipedia.org/wiki/Deformable_mirror.

[6] B. Park, *Two main categories of the deformable mirror. (a) Segmented and (b) continuous faceplate DM.* https://www.researchgate.net/figure/Two-main-categories-of-the-deformable-mirror-a-Segmented-and-b-continuous-faceplate_fig4_327848377.

[7]“Introduction to Adaptive Optics and Deformable Mirrors,” *Edmund Optics*. https://www.edmundoptics.eu/knowledge-center/application-notes/optics/introduction-to-adaptive-optics-and-deformable-mirrors/ .

[8] K. Hampson and M. Booth, “Calibration and Closed-Loop Control of Deformable Mirrors Using Direct Sensing.” https://aomicroscopy.org/dm-calibration-direct-sensing .

[9] “Eigen.” https://eigen.tuxfamily.org/index.php?title=Main_Page .

[10] G. Gregorčič, “The Singular Value Decomposition and the Pseudoinverse,” Apr. 11, 2001. https://www.cs.bgu.ac.il/~na181/wiki.files/SVD_application_paper%5B1%5D.pdf.