RESUMEN TRABAJO DE GRADO

AUTOR(ES): NOMBRES Y APELLIDOS COMPLETOS

NOMBRE(S): HENRY URLEY APELLIDOS: PORTILLA DELGADO

FACULTAD: EDUCACION, ARTES Y HUMANIDADES

PLAN DE ESTUDIOS: ARQUITECTURA

DIRECTOR:

NOMBRE(S): JUAN MANUEL APELLIDOS: VILLA CARRERO

NOMBRE(S): RAMON EDUARDO APELLIDOS: GALVIS CENTURION

TÍTULO DEL TRABAJO (TESIS):
APOYO LÍNEA DE SIMULACIÓN Y MODELIZACIÓN, GRUPO D_LAB, PRÁCTICA Y FORMACIÓN BÁSICA EN SIMULACIÓN DIGITAL PARA LA MODELIZACION DE ESTRUCTURAS AUTO ORGANIZABLES

RESUMEN

Este trabajo abarca la modelización de estructuras auto organizables desde la simulación de agregación de partículas de materia cristalina, este se llevó acabo laboratorio de diseño perteneciente al grupo investigación d_lab de la Universidad Francisco de Paula Santander.
Implementando el uso de herramientas de inteligencia artificial con la capacidad de organizar, analizar y generar predicciones sobre la agregación de partículas, del cual con los nuevos datos se realizó una visualización en simulador obteniendo estructuras.
Posteriormente, fueron analizados los modelos resultantes comparando la realidad experimentada y los resultados simulados para tener más detalle de las partículas agregadas resultando así la toma de decisiones sobre la modelización de estructuras auto organizables.
Para finalizar, partiendo de las características físicas y tridimensionales de piezas se diseñaron partículas con la capacidad de agregarse dando como resultado la formación estructuras con características similares a las cristalizaciones de la materia. Por lo tanto, la simulación en la modelización de estructuras auto organizables permite el diseño de piezas con capacidades similares al proceso de cristalización, pero se debe tener en cuenta que la simulación es solo la posibilidad de un suceso futuro por lo tanto comparar permite comprobar la validez entre las visualizaciones y la realidad para de esta modelizar estructuras auto organizables.

PALABRAS CLAVES: PARTICULAS, AGREGACIONES, MODELIZACION, AUTO ORGANIZACIÓN, SIMULACION.
CARACTERISTICAS:
PÁGINAS: 211 PLANOS:___ILUSTRACIONES: 78 CD ROOM: ___

APOYO LÍNEA DE SIMULACIÓN Y MODELIZACIÓN, GRUPO D_LAB, PRÁCTICA Y FORMACIÓN BÁSICA EN SIMULACIÓN DIGITAL PARA LA MODELIZACIÓN DE ESTRUCTURAS AUTO ORGANIZABLES

HENRY URLEY POTTILLA DELGADO

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

FACULTAD EDUCACIÓN, ARTES Y HUMANIDADES

PLAN DE ESTUDIO DE ARQUITECTURA

SAN JOSÉ DE CÚCUTA

2020

APOYO LÍNEA DE SIMULACIÓN Y MODELIZACIÓN, GRUPO D_LAB, PRÁCTICA Y FORMACIÓN BÁSICA EN SIMULACIÓN DIGITAL PARA LA MODELIZACIÓN DE ESTRUCTURAS AUTO ORGANIZABLES

HENRY URLEY PORTILLA DELGADO

Trabajo de grado modalidad de pasantía presentado como requisito para optar el título de Arquitecto

Director:

ARQUITECTO JUAN MANUEL VILLA CARRERO

Cotutor:

ARQUITECTO RAMON GALVIS CENTURIÓN

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

FACULTAD EDUCACIÓN, ARTES Y HUMANIDADES

PLAN DE ESTUDIO DE ARQUITECTURA

SAN JOSÉ DE CÚCUTA

2020

ACTA DE SUSTENTACION DE TESIS – mediadas por las TIC
PLAN DE ESTUDIOS DE ARQUITECTURA

Fecha: junio 17 de 2020

TITULO: APOYO LINEA DE SIMULACION Y MODELIZACION, GRUPO D_LAB, PRACTICA Y FORMACION BASICA EN SIMULACION DIGITAL PARA LA MODELIZACION DE ESTRUCTURAS AUTO ORGANIZABLES.

Presentado por: HENRY URLEY PORTILLA DELGADO Código 1500753

Modalidad: Pasantía, Investigación.

JURADO JAVIER ALBERTO MARIÑO DIAZ
FABIAN ALFREDO MENA USCATEGUI
JAVIER ANDRES LEMUS TORRES

DIRECTOR: JUAN MANUEL VILLA CARRERO
CO DIRECTOR RAMON EDUARDO GALVIS CENTURION

NOMBRE DEL ESTUDIANTE	CALIFICACIÓN	A. M. L.
HENRY URLEY PORTILLA DELGADO	5.0	LAUREADA

JAVIER ALBERTO MARIÑO DIAZ FABIAN ALFREDO MENA USCATEGUI

JAVIER ANDRES LEMUS TORRES

CARMEN XIOMARA DIAZ FUENTES
Directora Comité Curricular

Tabla de Contenido

Lista de Tablas

Pág.

Lista de Figuras

Pág.

Lista de Anexos

Introducción

La simulación es una herramienta de suma importancia en la actualidad, ya que a través de esta se puede realizar la toma de decisiones para un proyecto sin afectar directamente a este, con la simulación se podrá observar unos o varios sucesos que afecten al elemento simulado y a partir de los resultados tomar el camino de respuesta más eficiente. Sin embargo, se debe tener en cuenta que la simulación es una representación de una posible realidad, en donde solo se observa el elemento afectado por suceso específico, por lo tanto, la simulación más precisa es aquella cuyas especificaciones son más específicas con respecto a la realidad.

La simulación en el diseño igualmente permite observar cómo diferentes factores externos o de entorno alteran el elemento o modelo que está siendo sometido, también existe la posibilidad de generar simulaciones donde se puede observar la formación y creación de nuevos elementos partiendo de datos o elementos con información, es el caso de Andy Lomas, quien diseña modelos específicos y a través de uso de la simulación, le asigna características naturales de formación de células y tejidos animales o vegetales, y de esta manera se modelizan estructuras con la capacidad de dar forma actuando de manera similar a la natural a esto lleva el nombre de "formas celulares".

A través del uso de la simulación *Gediminas Kirdeiski*, modeliza estructuras basado en la agregación de volumetrías, esto se inicia modelando volúmenes individuales cuyas caras pueden ser controladas y seleccionadas libremente para la multiplicación y agregación usando las partes seleccionadas para finalmente estas ser materializadas, entonces, con la simulación se tiene la capacidad de modelizar nuevas estructuras partiendo de características naturales o a su vez

controlando las características físicas del elemento virtual a simular y de esta manera tener estructuras relevantes diseñadas y modelizadas desde la simulación como punto de partida.

Esta pasantía tiene como objetivo principal desarrollar estructuras auto organizables, partiendo de la simulación de agregaciones de partículas cristalinas, para dar respuesta al problema ¿cómo aplicar e implementar las simulaciones para la modelización de estructura auto organizables? Así mismo, busca apoyar el grupo *d_lab* en la implementación y aplicación de la simulación digital como medio para visualizar y analizar modelos de materia cristalina basados en la realidad. Donde se pretende aplicar tendencias de crecimientos y agregación de partículas con el fin generar hipótesis para modelizar estructuras cuyo comportamiento sea similar a lo observado en la experimentación. Para esto se establecieron tres etapas principales que se encuentran ligadas entre sí: proyección de datos, simulación de datos proyectados y modelización de estructuras auto organizables.

La proyección de datos consiste en el uso de datos basados en la agregación de partículas cristalinas, estos fueron analizados en un código *Python + anaconda* que posee la capacidad de estudiar y proyectar con el uso de métodos estadísticos y de aprendizaje automático arrojando una base de datos de posibles o futuras nuevas agregaciones de partículas cristalinas.

La simulación de datos proyectados, consiste en tomar los datos proyectados e ingresarlos en un simulador diseñado en *grasshopper + Rhinoceros* con la capacidad de interpretar grandes cantidades de datos e interpretarlos como puntos y volúmenes en el espacio que se agregan en base pasa el tiempo. Posteriormente con los modelos obtenidos se busca generar paralelos comparativos entre la realidad y a simulación para de esta manera determinar qué tan viable son los resultados y además observar los múltiples fenómenos presentes en la agregación de

partículas de materia cristalina y simulada para así determinar las bases en la modelización de estructuras auto organizables.

La tercera etapa es la modelización de estructuras auto organizables, consiste en la utilización de los fenómenos presentes tanto en la simulación como la realidad, de esta manera generar modelos con la capacidad de auto organizarse dentro de entornos dinámicos y finalmente modelizar estructuras relevantes que parten de características simuladas y presentes en la agregación de partículas cristalinas. Para modelizar estas estructuras se necesitó de la exploración de bases teóricas y antecedentes que hagan referencia a temas similares de esta manera fue posible obtener conceptos que guiaron a la modelización de estructuras auto organizables.

1. El Problema

1.1 Titulo

Apoyo línea de simulación y modelización, grupo *d_lab*, práctica y formación básica en simulación digital para la modelización de estructuras auto organizables.

1.2 Planteamiento del Problema

La simulación es un entorno u elementos artificiales que generan experiencias muy cercanas a la realidad por ejemplo en el caso de los videojuegos y simulaciones de entrenamiento entre otros, la experimentación del usuario cuenta un papel importante, ya que si este asimila la simulación y la hace parte de si, dejara de ser un elemento artificial para volverse momentáneamente en su realidad. Scheer, (2014) afirma “Hay muchas películas y libros entretenidos que exploran los peligros y las paradojas de un mundo en el que las simulaciones son indistinguibles del mundo real, donde la simulación y la realidad convergen” (p.31).

Por el contrario, en esta realidad el hombre ha convertido a la simulación en herramientas de entretenimiento, de entrenamiento o para la toma decisiones sobre probabilidades en escenarios que reflejan o se presentan en la realidad. La utilización de las simulaciones se ha implementado en múltiples campos donde la visualización de procesos es importante ya que no solo se puede observar un momento, sino que se pueden generar resultados de posibles futuros, que estos a su

vez influyen en la toma decisiones guiando a las respuestas más correctas y posiblemente más eficientes en los procesos y sistemas estudiados.

A pesar de la existencia de herramientas de simulación de fácil acceso se han seguido usando modelos basados en información concebida de manera estática cuya toma de decisiones solo se basa en información simplemente presentada en un momento especifico, por el cual el grupo *d_lab* emprendió un proyecto denominado: “Experimentación compleja de la materia y sus formas”, donde se pretende generar procesos emergentes de ideación. Actualmente los paradigmas de ideación en la Universidad Francisco de Paula Santander (UFPS), están atados a tecnologías de la construcción y en el mejor de los casos a tecnologías emergentes de fabricación en nuestro medio local, alejados de procesos de ideación vinculados a los crecimientos naturales de la materia.

Esta situación ha sido enfrentada en los talleres de diseño de la UFPS a través de un *workshop* llamado *Grown Matter* que tuvo como objetivo estudiar la agregación de minerales cristalinos en proximidad a partículas de semilla, como base para explorar el potencial de generación de formas y el crecimiento de partículas hacia la creación de objetos estructuralmente relevantes. En el *workshop* se contó con la participación como ponente de Ismael García ingeniero metalúrgico, persona encargada de explicar y exponer la experimentación con la cristalización de la materia.

Para iniciar con la experimentación se investigó e indago sobre los materiales con mayor reacción y potencial a ser cristalizados, los resultados fueron, azúcar, sal, sulfato de cobre, bórax y alumbre, estos materiales poseen una característica de baja toxicidad y de fácil asequibilidad, posteriormente una vez seleccionados los materiales se realizaron las primeras experimentaciones con el fin de observar la reacción de la materia al momento de cristalizarse, para de esta manera tomar decisiones sobre las próximas experimentaciones.

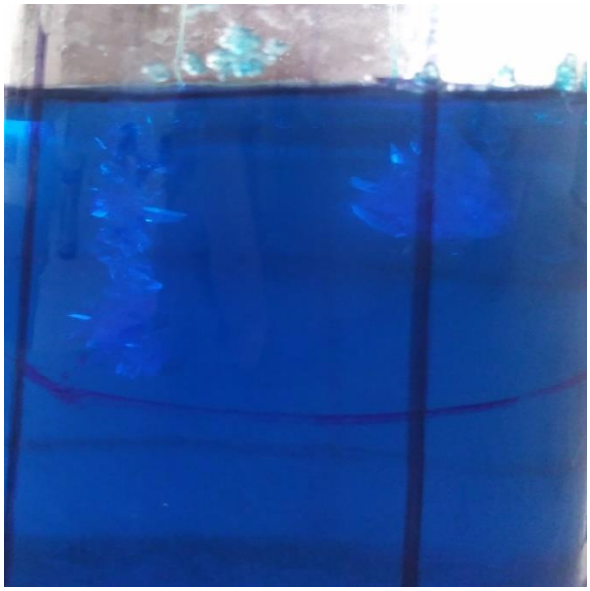


Figura 1 Primera experimentación, sulfato de cobre [fotografía]

Posterior a la primera experimentación se descartaron algunos materiales por factores como el tiempo extenso de cristalización o no son de fácil observación en la solución. Por el contrario, los materiales seleccionados se caracterizaron por sus cortos tiempos de cristalización y su fácil observación para la toma de datos (alumbre, bórax).

Figura 2 Materiales seleccionados, bórax**.** [fotografía]

El resultado de las primeras experimentaciones fue la aparición de partículas que se agregan, formando agrupaciones cristalinas en puntos específicos de los recipientes, de esta manera la observación de la cristalización de los materiales permite la toma de decisiones de cómo será mapeado este fenómeno de la manera más correcta posible. La cristalización de la materia es realizada disolviendo y sobresaturando el material seleccionado en agua, esta debe estar a temperaturas entre los 50 y 100 grados centígrados ya que el agua presenta mayor solubilidad a mayor temperatura, para realizar la mezcla correcta del material se recomendó el uso de la tabla de solubilidad.

Una vez la mezcla esta lista, se vierte sobre un recipiente, en este caso para el estudio de la formación de cristales, se implementó el uso de una caja de acrílico de 20x20x20 cm, cada una de sus paredes está trazada con una cuadricula que fue realizada en corte a laser con separaciones de 2 a 5 mm entre cada línea, al momento de la construcción de la caja se tuvo en cuenta que la cara donde se realizó el corte de la cuadricula apuntara al interior.

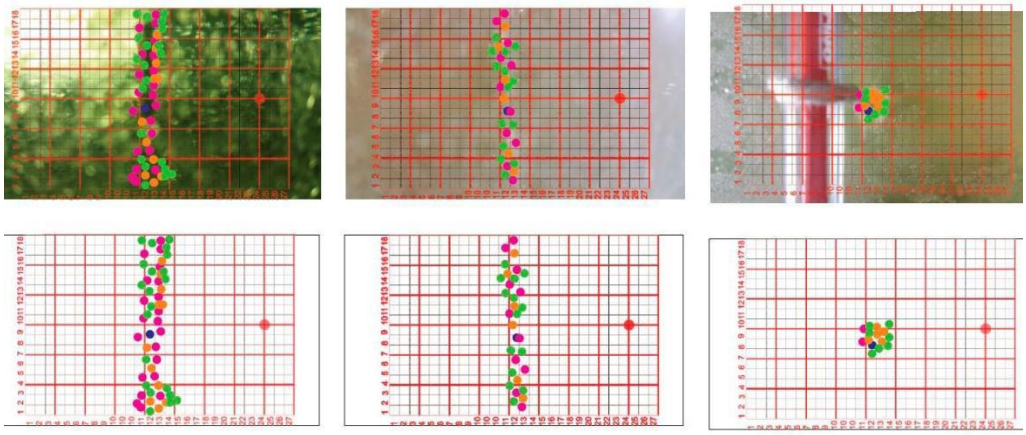
Figura 3 Diseño de caja, dimensiones de 20x20x20. [fotografía]

Ya con la solución vertida en la caja se introduce un hilo que se encuentra suspendido desde el exterior, una vez la mezcla se empieza a bajar su temperatura el proceso de cristalización empieza a ser evidente, las partículas se empiezan agregar en las paredes laterales de la caja, esto debido a las perturbaciones del corte laser, a esto le llamamos agregaciones 2D y además en el hilo se agregan partículas en todos los ejes a esto le llamamos agregaciones 3D.

Figura 4 Agregaciones, frontales agregaciones 2D y al fondo agregación 3D. [fotografía]

Las características del diseño de la caja en conjunto con la aparición de agregaciones cristalinas facilitaron el referenciado en el plano cartesiano y de esta manera se realizaron mapeos para el estudio de las agregaciones en lo que se tuvo en cuenta la dirección, el tiempo y las partículas. Estos mapeos permitieron la organización de los datos en lapsos de tiempo constante que oscilan entre los 5 a 15 minutos, esta información obtenida es representada en

coordenadas del plano cartesiano que posteriormente fueron digitalizada en tablas Excel para luego poder ser visualizada digitalmente en una simulación.



X_0	Y_0	Z_0	X_1	Y_1	Z_1	X_2	Y_2	Z_2
12	8,5	8,5	11	8,5	3	13	8,4	3
11	8,5	2	11,5	9,2	16,5	13	9,3	17
13	9	15	14	9,5	2	12	9,7	2
13	9,5	10	14	7,3	2	15	10	2
12	7,5	10	12,5	8,6	4	13	8,2	6
11	8,5	4	12	9	5	12	9,5	7
13	9,1	8	12	9,8	3	11,5	9,2	4
13	9,9	3	11	8,5	8,5	13	8,6	9
14	10	3	13	7,5	13	13	9,2	14,5
13	8,7	7	12	9	10	12	10	11,5
13	7,9	11	11	8,4	10	13	8,1	10
12	9	6	14	9	14	13	8,8	14
13	8,5	10	12	10,2	13	11	8,6	12
13	9,1	16	13	8	11	12	8,2	14

. ***Figura 5*** Arriba mapeos de agregaciones 3D, abajo organización de datos de mapeo.
Fuente: [captura] Elaborado por Jorge Hernández y Camilo Ramones

La simulación fue diseñada en conjunto con Juan Manuel Villa, Pablo Rey, Henry Portilla y Miguel Ángel Quintero, con el uso herramientas digitales como el programa *Grasshopper* se construyó la simulación y se visualizó en *Rhinoceros*, la simulación consiste en la inserción de las tablas Excel resultantes de los mapeos y posteriormente se visualiza la aparición de las partículas ya sean 2D o 3D en lapsos de tiempos más cortos y de esta manera comprar la simulación y la realidad. Además, el participante Henry Urley Portilla realizó modificaciones que permitirán la inserción de grandes cantidades de datos a la simulación.

La simulación tiene la función de interpretar los datos recolectados en coordenadas x, y, z que son representados en el visualizador como puntos en el espacio que surgen con el paso del tiempo que es equivalente al tiempo y orden en el que las agregaciones cristalinas se forman en

la experimentación. De la simulación se obtienen modelos tridimensionales que pueden ser comparados por similitud a lo visualizado durante la experimentación, de este modo se puede comprobar si los datos recolectados están correctamente georreferenciados.

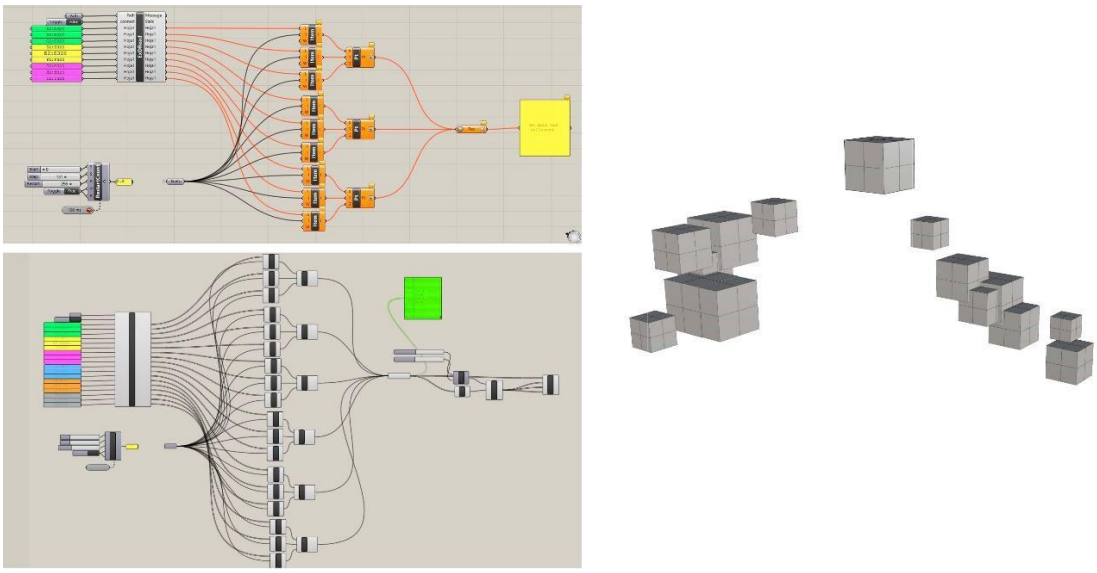


Figura 6 Izquierda superior simulación original, izquierda inferior definición modificada, derecha resultados de simulación datos de experimentación. [captura]

Sin embargo, el objetivo del *workshop* es la orientación de la materia, basado en esto, se necesita poder proyectar los datos de la experimentación para de esta manera visualizar a mayores cantidades de lapso de tiempo y espacio el crecimiento en la agregación de las partículas para esta manera poder observar la posible orientación de las agregaciones cristalinas, por esto a cada uno de los grupos participantes del workshop se le pidió que explorara maneras de proyectar y crear tendencias basados en los datos de la experimentación.

El método de proyección más usado por los participantes del *workshop* fue las proyecciones por tendencia presentes dentro del programa *Excel*, que consiste en tomar el último número de la columna y proyectarlo basado en los datos anteriores a este. Con los nuevos datos obtenidos de la proyección se realiza la simulación del cual se obtienen una visualización irreal de agregación

de partículas cristalinas ya que las partes se empiezan a generar de manera unidireccional esto se debe a que el tipo de proyección usado es proyección estadística lineal y las agregaciones cristalinas poseen una agregación de partículas no lineal.

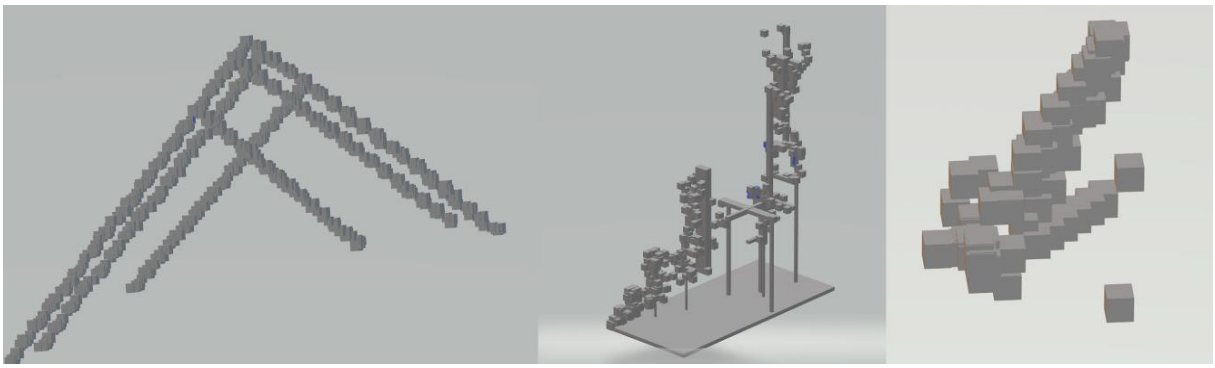


Figura 7 Proyección análoga
Fuente: Izquierda proyección Henry Portilla y Camila Araque, centro proyección Cristina Lizcano y Jordin Suarez, derecha proyección John Mantilla y Sebastián Ortega.

El grupo de trabajo conformado por Miguel Ángel Quintero y Juan José Conde presentaron una proyección basada en programación en lenguaje Python usando biblioteca del programa *Anaconda*, esta proyección dio como resultado proyecciones de hasta 3000 datos basados en los obtenidos de la experimentación y al ser visualizados en la simulación dio como derivación proyecciones realmente cercanas a la realidad siendo este un gran aporte al conocimiento del grupo y el *workshop*. El grupo de trabajo no tenía fiel conocimiento del funcionamiento real e interno del código *Python*, ya que este código fue escrito por el físico Joseph Fabricio Vergel que fue contactado de manera independiente por este grupo, lo que provocó que no se tuviese la información y el conocimiento para realizar próximas proyecciones de los datos obtenidos en las experimentaciones.

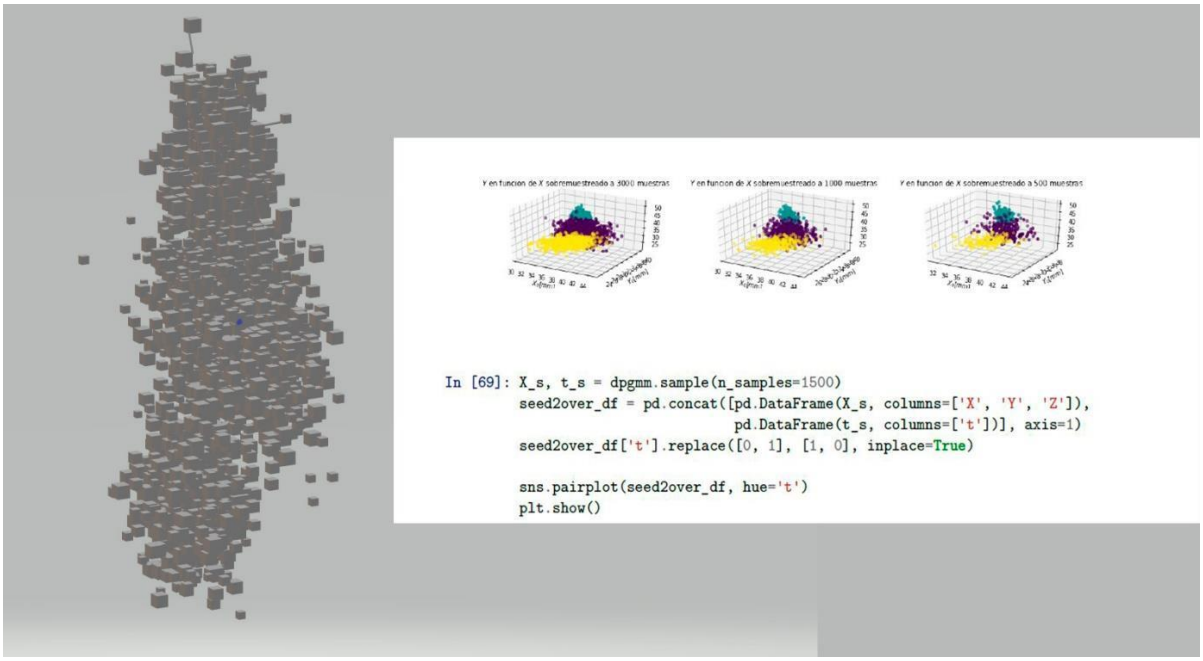


Figura 8 Proyección y tendencia basado en inteligencia artificial de *Python*.
Fuente: [captura] Elaborado por Miguel Ángel Quintero y Juan José conde

Para finalizar se realizó la materialización de los resultados obtenidos de las proyecciones en la simulación sin importar si su resultado era lejano o cercano a la realidad, los modelos obtenidos durante la simulación fueron exportados a formatos *stl* para su posterior impresión en 3d, obteniendo de esta manera las estructuras físicas derivadas de las proyecciones lineales. Por tanto, la presente propuesta de proyecto de pasantía, pretende ampliar el proceso, continuando con la simulación de la materia cristalina basados en datos obtenidos de la experimentación para el alcance de modelos cercanos a la realidad, usando herramientas como *Python* principalmente, cuyos resultados serán analizados y simulados de los cuales se obtendrán modelos próximos a la realidad de las agregaciones de partículas y a partir de esto modelizar estructuras relevantes que funciones similar a la agregación de partículas.

1.3 Formulación del Problema

En virtud de lo planteado, el presente proyecto de pasantías para el grupo de investigación *d_lab*, aborda el siguiente interrogante:

¿Cómo aplicar e implementar las simulaciones para la modelización de estructura auto organizables?

1.4 Justificación

El presente trabajo de pasantía se enfocará en apoyar al grupo *d_lab*, con la aplicación e implementación de la simulación digital que se da desde el desarrollo de esta y se trabajará de la mano con ,apoyo línea de simulación y modelización, grupo *d_lab,* práctica y formación básica en experimentación y mapeo de estructuras de agregaciones cristalinas, que a través de datos obtenidos de la experimentación se busca generar hipótesis de agregaciones y crecimiento, para posteriormente observarlos a través de las simulaciones.

La simulación aparece con la necesidad de generar modelos cercanos a la realidad y así mismo utilizar los datos con el fin de generar tendencia de crecimiento, para poder visualizar la agregación de cristales en rangos de tiempo y espacio mucho más amplios a los posibles en la realidad. Consecuentemente, para la creación de esta realidad simulada debe previamente ser desarrollada, que en este caso se contará con el uso de herramientas de lenguaje de programación grafico (*Grasshopper*), las bibliotecas de análisis de datos usados en el lenguaje de programación (*Python*) y una interfaz para la visualización del modelo que simula la realidad (*Rhinoceros*).

La intención es poder usar la realidad simulada con el fin de comprender y entender la naturaleza de la materia cristalina que permitirá la toma de decisiones para la orientación de esta y así mismo lograr la formación de estructuras relevantes. Entonces la implementación de la simulación se vuelve en un elemento de suma importancia para la predicción de posibles formaciones cristalinas, ya que a través de la inserción de datos se creará modelos donde se podrán observar la posibilidad de crecimiento, el tiempo y la dirección en las agregaciones de la materia cristalina.

Además, con el uso de programas alternos se podrá generar tendencias de agregación y así mismo visualizar la materia en rangos de formación más amplios de espacio, tiempo y agregación para comparar los resultados de la simulación con la realidad experimentada para así comprender los fenómenos de la agregación en los materiales cristalinos y finalmente modelizar la agregación de partículas con la posibilidad de formación de estructuras relevantes.

1.5 Objetivos

1.5.1 Objetivo General. Desarrollar estructuras auto organizables partiendo de la simulación de agregaciones de partículas cristalinas.

1.5.2 Objetivos Específicos.

Proyectar los datos obtenidos de la experimentación con materia cristalina realizado durante el desarrollo del *workshop*.

Simular los datos obtenidos de la proyección en el simulador desarrollado en *grasshopper + Rhinoceros.*

Modelizar estructuras basadas en la simulación.

1.6 Alcances, Delimitaciones y Limitaciones

1.6.1 Alcances. Este proyecto abarca el análisis y visualización digital de datos obtenidos en la experimentación y la observación de agregación de partículas, con los datos obtenidos se busca realizar proyecciones y tendencias para crear hipótesis de agregación sobre la materia.

Al mismo hacer uso de la simulación diseñada durante el workshop, donde los resultados de las tendencias puedan ser aplicados con el fin de poder ser visualizados, para obtener modelos que puedan ser comparados con la realidad y de esta manera se tomen decisiones al momento de controlar y direccionar cristales. Además partiendo de los resultados simulados, modelizar estructuras que puedan ser comparadas con la realidad de las agregaciones cristalinas ya pueden llegar a poseer características de agregación.

1.6.2 Delimitaciones.

1.6.2.1 Delimitación Espacial. La pasantía y el proyecto se llevará a cabo dentro de las aulas pertenecientes al grupo de investigación *d_Lab* que corresponden a SF101 Y SF102 dentro de la universidad Francisco de Paula Santander.

1.6.2.2 Delimitación Temporal. Las pasantías en conjunto al cumplimiento de los objetivos cuentan con un lapso de cuatro meses que serán ocupados durante el segundo semestre del 2019 y parte del primer semestre de 2020, este tiempo será repartido para el cumplimiento de los objetivos y actividades.

1.6.3 Limitaciones.

Déficit de datos que limiten la proyección de estos.

Fallas y errores en la configuración de definiciones para la simulación.

Errores de escritura en el código *Python*.

2. Marco Referencial

2.1 Antecedentes

En el siguiente recuento se tomaron antecedente como guía base para el desarrollo de esta propuesta.

Nombre: *Formas celulares: una exploración artística de la morfogénesis*

Autor: Andy Lomas

Año: 2014

Lugar: SIGGRAPH Studio (Vancouver)

Objetivo: Implementar una simulación para la formación de estructuras de manera emergente, además crear avatares de células diseñadas digitalmente que poseen información o las reglas del crecimiento de células naturales pero esta vez controlando los rangos de agregaciones y de espacialidad de esta manera así explorar y crear múltiples formas similares encontradas en la naturaleza pero que son representadas y creadas por los medios artificiales. Estas simulaciones fueron logradas a través de la utilización de lenguaje de programación C++ y visualizadas con CUDA.

Síntesis: creación de simulaciones basado en informaciones o patrones de formación en las células en el mundo natural que luego las implementa al mundo digital logrando resultados de visualización de estructuras entre lo natural y artificial.

Figura 9 Lomas, A. (2014) estructura con control de agregaciones
Fuente: [modelo] recuperado de "http://andylomas.com/extra/andylomas_paper_cellular_forms_aisb50.pdf"

Conclusiones: La utilización de información real e implementada dentro de las simulaciones.

El control de la información dando rangos de agregación.

La utilización de múltiples lenguajes de visualización y programación para la creación de simulaciones.

Nombre: *Agregación y modelado basado en gráficos.*

Autor: Gediminas Kirdeikis, Petras Vestartas

Año: 2017

Lugar: Taller en la Academia de las Artes de Vilnius (VAA). Lituania.

Objetivos: creación, desarrollo, control total de partículas simuladas y parametrizadas para la formación de estructuras.

Síntesis: se creó una programación que permite la creación de cualquier tipo de partícula parametrizada para posteriormente ser completamente simuladas y controlada permitiendo así la

alteración en la morfología de las partículas y la alteración del comportamiento respecto a las demás partículas que se van agregando con el fin de crear estructura controladas. Estas simulaciones fueron diseñas y programadas con el editor de lenguaje de programación grafico grasshopper en conjunto de un plugin denominado fox.

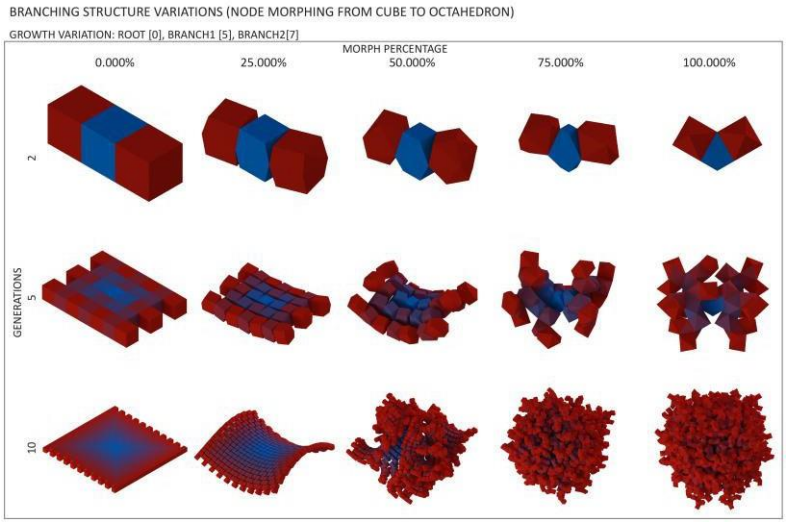


Figura 10 Kirdeiski, G. (2017) de cubo a octaedro y sus agregaciones.
Fuente: [simulación] recuperado de "https://gkirdeikis.wordpress.com/portfolio/research-title-placeholder-ii/"

Conclusiones: La simulación permite el control de los elementos en particular o de múltiples elementos al tiempo, basados en patrones y características físicas asignadas.

Establecimiento de reglas propias o basada en elemento reales para el crecimiento de morfologías únicas y formación de nuevas estructuras.

La simulación permite reinventar la arquitectura desde nuevos métodos.

La utilización de programas de diseño

Nombre: *Modelo para la simulación de sistemas de multi-agentes robóticos en Python.*

Autores: Andrés Camilo Jiménez, John Petearson Anzola, Giovanny Mauricio Tarazona, Sandro Javier Bolaños.

Año: 2017

Lugar: Universidad Distrital Francisco José de Caldas, Bogotá, Colombia

Objetivo: Diseño de un modelo de simulación programada en *Python* de robots multi-agentes.

Síntesis: "En la etapa de diseño de Sistemas Multi-Agentes Robóticos, la validación de algoritmos y la verificación del modelo cinemático inverso y directo, es importante para la detección de problemas o errores antes de implementarlos en el agente físico" (Jimenez, Anzola, Tarazona, & Bolaños, 2017), se realizó la inserción de códigos que reaccionara a la proximidad del robot a superficies y además estos códigos permitieron a través de la simulación en Python el aprendizajes en la toma de direcciones para buscar la salida del laberinto. El modelo de robot poseía características de un robot real estas fueron dadas a partir de la parametrización del modelo a través del lenguaje de programación de Python en alianza con bibliotecas de análisis, bibliotecas físicas y matemáticas.

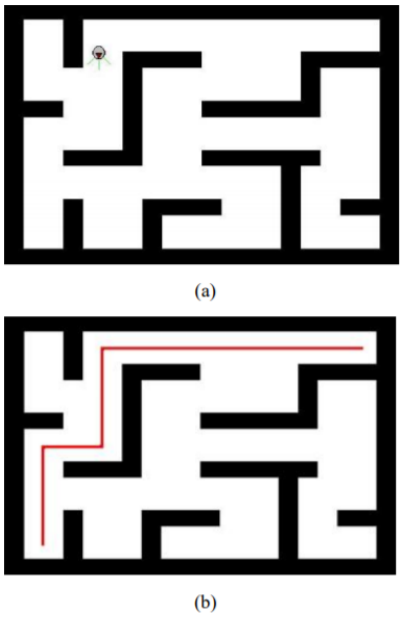


Figura 11 Simulación de laberinto
(a) Modelo ubicado en el interior del laberinto, (b) Resultado del recorrido al solo evadir obstáculos por la izquierda.

Fuente: modelo_para_la_simulacion.pdf

Conclusiones: La utilización de programas de lenguaje de programación para la creación de modelos más próximos a la realidad.

La precisión de los datos simulados para obtener resultados que permitan ser implementados en la realidad.

Nombre: *Laboratorio de auto-ensamble*

Autor: Skylar Tibbits

Año: 2012-2015

Lugar: MIT (Massachusetts Institute of Technology)

Objetivo: Generar nuevas tecnologías y métodos constructivos basados en el auto ensamblado o autoconstrucción.

Síntesis: El auto ensamblado consiste los elementos individuales desordenados ubicados en un espacio determinado cuya función es auto ordenarse o agruparse para la formación de estructura. Los elementos son sometidos a movimientos y oscilaciones en busca de un estado final agrupado. Para la unión y ensamblado usan imanes como conectores.

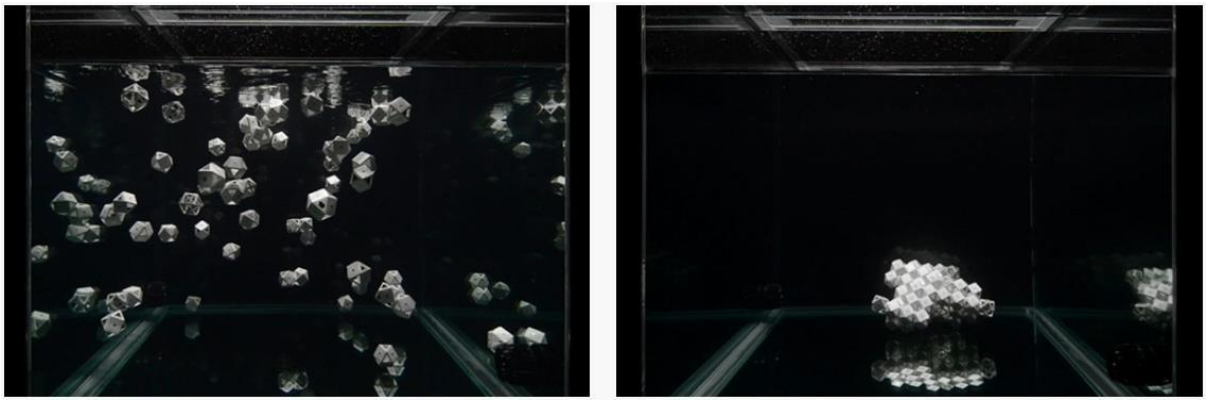


Figura 12 Auto ensamblaje en entornos complejos
Fuente: Tibbits A, Zuniga B, McKnelly C, Papadopoulou A. (2015)**.** [EXPERIMENTACION] recuperado de “https://selfassemblylab.mit.edu/fluid-lattices”

Conclusiones: Formación de estructuras a partir de elementos individuales que se unen para dar forma.

La autoconstrucción como modelo emergente para el futuro de la construcción.

Fabricación auto evolutiva en el que el material se organiza según las dinámicas del entorno.

Nombre: *Self-assembly gets physical*

Autor: Arthur Olson

Año: 2015

Lugar: Nature Nanotechnology

Objetivo: A través de la comprensión, análisis y experimentación de elementos naturales, crear modelos de entendimiento de los procesos de formación natural.

Síntesis: Diseño de un modelo tridimensional es logrado basando en la formación del polio virus y a través de la observación y medición de este, con el objetivo de enseñar a sus estudiantes de un método más práctico de explicar la formación de los virus. Para la construcción de este modelo se necesitó diseñar un modelo tridimensional que facilita los modelos tradicionales usados en la década de los 70, además se desarrolló un método constructivo que permitiese ser armado sin la intervención directa de la mano del hombre eso logrado a través de la imantación. Pero no solo busca ver como se construye un virus sino analizar la probabilidad de ensamblado de 12 piezas modulares dando como resultado una combinación 10 a la 15 en unos tiempos que oscilan entre 10 a 30 minutos, y si cada pieza fuese individualmente única la autoconstrucción y auto ensamble tardaría años.

Figura 13 Olson, A. (2005) construcción de polio virus
Fuente: [modelo] recuperado de "https://www.nature.com/articles/nnano.2015.172"

Conclusiones: Método de interconexión y ensamblado de piezas que son individuales que conforman un todo.

El análisis de procesos naturales para la generación de modelos que simulen una realidad o que los mimetice.

Modulado de piezas y ensamble para mayor probabilidad de formación de modelos únicos con piezas individuales.

2.2 Bases Teóricas

La recopilación teórica para este proyecto de pasantías está encaminado a evidenciar

postulados sobre la importancia aplicación de la simulación y modelización como herramienta de visualización previa a la materialización o toma de decisiones.

Simulación

La simulación es una herramienta en la cual pueden proyectarse, evaluar y contemplar nuevos procesos o ya existentes, lo que permite experimentar con los procesos desde una perspectiva sistemática para un mejor entendimiento de causas y efectos, pero no solo esto también se pueden llegar a lograr predecir situaciones, si no que de esta manera comprender y analizar e incluso llegar a visualizar el futuro del proceso o sistema que se está siendo sometido a la simulación (Fullana & Urqia, 2009).

Podría decirse que la experimentación puede hacerse directamente aplicando sobre el modelo a estudiar, pero esto provocaría directamente riesgos para la realidad del modelo pues se generarían riesgos en costos, seguridad entre otros (Tarifa ,2001). El caso es contrario cuando la experimentación es realizada desde la simulación pues los resultados efectuados de esta conllevan a la toma decisiones en la realidad de esta manera ahorrando la pérdida o daños en el modelo a estudiar.

Cualquiera que sea el motivo por el que se desea implementar la simulación es importante la creación de un modelo y que este sea lo más fiel posible a la realidad, para que los resultados permitan una visualización más real de un futuro comportamiento. García y Ortega (2006) afirman que:

> La necesidad de simular aparece cuando el sistema o modelo a estudiar resulta excesivamente complejo y esta puede tener básicamente dos causas:
>
> El sistema presenta comportamientos no lineales que producen de esta manera que sea imposible un análisis y resolución analítica o una análoga.

> La aparición de fenómenos aleatorios donde su análisis o predicciones son casi imposibles y esto lleva a concurrir a el uso de términos de probabilidad.” (p. 5).

De acuerdo a lo anterior, las simulaciones permiten visualizaciones de comportamientos que puede ser complejos de visualizar a largos plazos, pero solo esto, sino que permite la creación de proyecciones y pronósticos sobre los modelos e incluso fenómenos a estudiar o experimentar. En las simulaciones se puede tener un mejor control de los experimentos y de esta manera estimar comportamientos, pero las simulaciones solo pueden dar como resultado una estimación a la vez basado en unos parámetros establecidos lo que con lleva a una múltiple ejecución de la simulación en distintos parámetros o posibilidades que se desees experimentar sobre el modelo.

Modelización

La modelización es “formalizar un fenómeno natural, organizacional o técnico” (Fernández, s.f, p. 2). Las modelizaciones deben poseer una descripción muy detalla del elemento o fenómeno a modelar, se realiza con el fin de facilitar la compresión sobre algo específico, para esto se debe realizar una abstracción o esquematización, se deben omitir detalles que no sean importantes en la visualización del modelo para que la comprensión de este sea muy simple.

Existen tres tipos de modelización y con cada uno es una posibilidad diferente de representar la realidad, un sistema, procesos, fenómenos, entre otros. García & Ortega (2006) hablan sobre estos tipos de modelización que son:

> Modelos físicos: son modelos tangibles que poseen características del objeto a estudio, un ejemplo de estos es la realización de maquetas a escala.
>
> Modelo matemático: se representan por medios lógicos y cuantitativos de las variables del estado del modelo, de esta forma este puede ser estudiado y además permita la modificación de sus características numéricas, según la complejidad del

estudio este puede ser analizado analíticamente, pero si el caso es contrario presentando complejidad lo más aconsejable es recurrir a la simulación.

Modelo simbólico matemático: este modelo surgió con la aparición de las herramientas de modelado digital. “Los modelos simbólicos matemáticos mapean las relaciones existentes entre las propiedades físicas del sistema que se pretende modelar en las correspondientes estructuras matemáticas” (Domínguez & et al., 2008. p. 7).

Cualquiera que sea el tipo de modelo que se esté creando y según la necesidad de estudios estos anteriores pasaran por tres etapas de tipologías de modelos como son; modelos de objetos, se refiere a la descripción estructural del modelo en estática o sea el estado actual; modelo dinámico, se refiere y describe al modelo y su comportamiento con el paso del tiempo; y, el modelo funcional, se refiere a las características dadas por las transformaciones aplicadas al modelo (Domínguez & et al., 2008).

La modelización se basa en la creación de modelos tangibles o intangibles pero que siempre se ve una realidad representada de la manera más comprensible y con la información necesaria para el estudio y este será realizado desde las técnicas análogas o de simulación.

Regresión Lineal

Es una técnica utilizada para estudiar la relación entre variables, con este tipo de técnica es posible generar ecuaciones con fines predictivos. Consiste en trazar una recta que se encarga de analizar una nube de puntos cercana a esta con el fin de generar un análisis de correlación de variables. Existen dos tipos de regresiones lineales como lo son regresión lineal simple y regresión lineal múltiple.

Regresión Lineal Simple

Consiste en el trazo de una recta dentro de un plano cartesiano bidimensional, para realizar la selección de la recta se tienen en cuenta y se realiza un análisis de recta en relación a la nube de puntos, pero solo una de estas rectas es la más viable para realizar la sección se tomará en cuenta la recta en la que los puntos tengan menor distancia a la recta en un sentido vertical.

Regresión Lineal Múltiple

Cuando la nube de puntos se encuentra dispersa de manera tridimensional o multidireccional se acude a la regresión lineal múltiple este método se encarga no solo de analizar los datos a través de una recta sino de un plano multidimensional que es trazado más óptimamente dentro de la nube de puntos (Marín, 2014).

2.3 Marco Conceptual

Auto ensamblaje: El auto ensamblaje es un término que se usa constantemente en la química para referirse al método potente y natural en la creación de estructuras complejas previamente definidas presentes principalmente a escalas micrométricas. Estas estructuras son formadas a partir de las moléculas más pequeñas, estas unidades se agregan para formar estructuras o formas establecidas por la naturaleza. El diseño de autoensamblajes artificiales se encuentra actualmente en etapas iniciales ya que para lograr unir partes de una manera preestablecida se requiere tener en cuenta las condiciones de entorno a que son sometidas las partículas a auto ensamblar (Ferrer & et al, 2005)

Auto organización: Es un proceso en el que de manera espontánea unas partes individuales conforman algún tipo de forma estructura global. La auto organización no debe ser controlado ni dirigido, este mismo busca dar sus propios resultados finales generales, partiendo de la individualidad. Este proceso se da en diferentes fenómenos naturales como la cristalización y la emergencia en patrones, en la aglomeración de humanos y animales, en todos los casos son seres o partes individuales que poseen ciertas características que al ser juntados pueden formar diferentes conjuntos influidos principalmente por su entorno (autoorganizacion, 2013).

Grasshopper: "*Grasshopper* es un editor de programación visual desarrollado por David Rutten en *Robert McNeel & Associates*. Como un *plug-in* para *Rhino3D*. *Grasshopper* está embebido en el robusto y versátil entorno de modelación utilizado por profesionales de diversas y variadas industrias creativas, tales como arquitectura, ingeniería, diseño de producto y otras más. En conjunto, *Grasshopper* y *Rhino* nos ofrecen la oportunidad de definir control paramétrico preciso sobre nuestros modelos, la capacidad de explorar *workflows* de diseño generativo, así como una plataforma para desarrollar una lógica de programación de alto nivel" (MODELAB, 2015).

Lenguaje de programación gráfico: "Se define como un software que permite el acceso de datos remotos de un proceso y también permite, utilizando las herramientas de comunicaciones necesarias, el control del mismo. Atendiendo a esta definición podemos deducir que no se trata de un sistema de control, sino de una utilidad software monitorización o supervisión que realiza la tarea de interfase entre los niveles de control" (Tamami, 2015).

Pronostico: “El término pronóstico refiere a aquel conocimiento anticipado de lo que sucederá en un futuro mediante ciertos indicios, señales, síntomas, intuiciones, estudio, historia previa, entre otros, que se suceden cumpliendo una función de anuncio” (Ucha, 2010)

Python: “*Python* es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad” (Alvarez, 2013).

Rhinoceros: “*Software* para diseño industrial donde puedes crear diseño de productos ya sea de consumo, joyería, enseres, automotriz, aeroespacial. *Rhinoceros* incluye un modelador de superficies tipo NURBS especializado para diseños de formas libres y orgánicas, el programa propone eliminar los cuellos de botella que se puedan presentar en el flujo del proyecto de diseño” (CAID,s.f).

Tendencias: “La tendencia es una corriente o preferencia hacia determinados fines u orientar en cierto rumbo” (Perez, definicion.de, 2012).

Virtual: “El concepto, de todas formas, está actualmente asociado a lo que tiene existencia aparente, opuesto a lo real o físico” (Perez & Gardey, definicion.de, 2013).

3. Diseño Metodológico

3.1 Tipo de Investigación

Este proyecto de apoyo posee un enfoque de investigación tipo aplicada, ya que a través de la aplicación de técnicas y tecnologías de simulación se busca comprender la materia cristalina y de esta manera generar poder visualizar de un modo muy cercano a la realidad, las posibilidades y partiendo de esta poder modelizar estructuras similares a lo observado en la experimentación.

> La investigación aplicada tiene por objetivo la generación de conocimiento con aplicación directa y a mediano plazo en la sociedad o en el sector productivo. Este tipo de estudios presenta un gran valor agregado por la utilización del conocimiento que proviene de la investigación básica (Lozada, 2009).

Inclusive si se habla de una investigación según la manipulación de las variables, se puede decir que se realizó un investigación experimental ya que con los datos dados de las tendencias y la experimentación se podrán usar a favor para la comparación de los resultados y de esta manera diseñó y materializó estructuras basadas en la información adquirida por la simulación y la experimentación.

> La Manipulación de las Variables se centra en la manera como se desea controlar o no las variables… Investigación experimental: Se manipula una o varias variables independientes, ejerciendo el máximo control. Su metodología es generalmente cuantitativa" (Cardenas, 2010).

3.2 Fases o Etapas

Revisión del Estado de la Cuestión

Esta etapa se desarrolla a lo largo de toda la realización del proyecto en modalidad de pasantías en la que se realizara investigaciones para explicar los fenómenos observados en la experimentación y el cómo la simulación puede reforzar los términos encontrados. También se investigarán referente que permitan guiar en la modelización de estructuras basados en la agregación de partículas. Esta revisión también permitirá estudiar el código *Python*, sus funcionamientos y su reproducción para obtener la proyección de los datos obtenidos durante la experimentación en el *workshop*.

Proyectar las Tendencias de Crecimiento de la Materia

Con el uso de la información obtenida de las experimentaciones con la materia cristalina se realizarán análisis para hallar patrones y tendencias de crecimiento o agregaciones. Esta etapa se logrará a través de la inserción de datos en plataformas para el análisis de datos tales como *Excel* o *Python* con alianza en bibliotecas de información estadística y física. Los datos resultantes del análisis y la tendencia permitirán una visualización de crecimiento a través del paso del tiempo.

Simulación

En esta etapa se busca la visualización de los datos obtenidos de la experimentación lo más cercanos a la realidad y de esta misma forma poder observar en modelo los datos obtenidos de las tendencias. Esta visualización es posible a través del uso de simulación diseñada durante el

desarrollo del workshop que permita la inserción de datos en forma de coordenadas de manera ordenada en tiempo, para la observación de un modelo dinámico en el espacio y el tiempo.

Diseño y Modelado de Estructuras

En esta etapa se busca que a través de los resultados visualizados en la simulación se puedan generar piezas que permitan la construcción de estructuras similares y posteriormente estas piezas puedan ser comprobadas a través de la experimentación o la realización de ejercicios con estos nuevos elementos.

Materialización de Estructuras

En esta etapa se ejecutarán los ejercicios con las nuevas piezas para verificar y observar la creación de estructuras relevantes basadas en fenómenos similares a los observados en la agregación de partículas en cristalización de la materia.

3.3 Hipótesis

Se obtienen modelos de estructuras auto organizables con características derivadas de la simulación y la agregación de partículas, además en estos modelos se pueden observar los fenómenos similares a la cristalización.

4. Administración de la Investigación

4.1 Recursos Humanos

Este proyecto de apoyo contó con la participación de:

Henry Urley Portilla delgado, como autor de este trabajo

Juan Manuel Villa Carrero, como director del proyecto de apoyo

Ramon Galvis Centurión, como cotutor en el proyecto de apoyo

4.2 Recursos Institucionales

Como recursos institucionales se requiere el uso del laboratorio de diseño *d_lab* ubicando en las aulas SF101 y SF102 ubicados dentro de la universidad Francisco de Paula Santander.

4.3 Recursos Materiales

Para la realización de este proyecto se requieren los siguientes elementos materiales:

Computador de alto rendimiento y que en su software tenga instalado *Rhinoceros* + *grasshopper* principalmente.

Una impresora 3d, para materialización de modelos

Filamento PLA para impresión 3d

4.4 Cronograma de Actividades

Mes, 2019	**1**				**2**				**3**				**4**			
Actividad	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
1. Revisión estado cuestión																
2. Proyectar las tendencias de crecimiento de la materia en la simulación.																
3. Desarrollo simulación digital (Grasshopper, Python).																
4. Diseño y modelado de estructuras relevantes de acuerdo a los resultados obtenidos																
5. Materialización de las estructuras obtenidas en las simulaciones																
6. Sistematización																

Tabla 1 Cronograma

5. Resultados

5.1 Resumen Técnico

Este proyecto de pasantía pretendió exponer de modo detallado todas las actividades y los resultados obtenidos, consiguientemente, en el desarrollo de este proyecto se llevaron a cabo tres principales etapas, para dar respuesta al planteamiento del problema y adicionalmente cumplir con los objetivos trazados. Cada etapa tiene un objetivo a cumplir, que se entrelaza a las etapas, para de esta manera dar respuesta al objetivo general, estas son: proyección de datos, simulación de datos proyectados, modelización de estructuras auto organizables.

Como primera etapa la proyección de datos está basada en la información recolectada de experimentaciones y mapeos que fueron realizados durante el *workshop*. Para realizar las proyecciones se usaron herramientas de predicción de datos generados en lenguaje de programación *Python*.

La segunda etapa, da uso a los resultados obtenidos de la proyección son visualizados en una simulación diseñada en *Grasshopper* con el fin visualizar las posibilidades de agregación de partículas y compararla con la realidad experimentada para posteriormente estudiar los fenómenos en el proceso de la agregación de materia cristalina.

Y como tercera etapa, modelizar y materializar la agregación de partículas y sus fenómenos en volúmenes o formas que repliquen la cristalización para obtener un sistema dinámico de adición de partículas capaz de auto organizarse para la creación de estructuras basado en una simulación derivada de un proceso químico natural.

5.2 Cumplimiento de Objetivos

Para dar con el cumplimiento de la pasantía de sebe tener claro primordialmente cual es el tema a tratar dentro del trabajo realizado posterior a esto se deben plantear unos objetivos que en el caso de este trabajo son tres específicos y uno principal que delimitan los alcances dentro del proyector. El primer objetivo consta de la proyección de datos a través del uso de herramientas de programación para el análisis de datos organizados de manera no lineal en el tiempo y el espacio. El segundo objetivo consta en tomar los resultados obtenidos del análisis y proyección de datos para simularlos de esta manera obtener estructuras de posibles agregaciones de partículas cristalinas y así comprar estos con la realidad.

Esto, con el fin de concluir en los fenómenos y lo sucesos al momento de cristalizar la materia, para de esta manera generar modelos físicos que poseen la capacidad de auto organizarse modelizando nuevas estructuras de manera similar a la cristalización. Cada uno de los anteriores objetivos anteriores fue desarrollado de manera consecuente u ordenada para así dar con el correcto desarrollo de este proyecto de pasantías. Para revisar el cumplimiento de los objetivos y actividades revisar cronograma en anexos A.

5.3 Cumplimiento de la Metodología

Basados en los datos obtenidos de las experimentaciones realizadas durante el workshop, este proyecto se apoya en la aplicación de herramientas digitales con la capacidad de predecir esto es logrado a través del uso de un código de algoritmos en *Python* enlazados a bibliotecas de anaconda que permiten el análisis de datos y en respuesta a esto se obtienen datos que

representan la posibilidad de orientación y agregación de partículas al ser expuesto a mayor cantidad de tiempo. Los datos nuevos obtenidos pueden ser visualizados con el uso de una simulación desarrollada en *grasshopper + Rhinoceros*, esta posee la capacidad de interpretar los datos proyectados e interpretarlos en puntos en el espacio para de esta manera ver el posible crecimiento y como se agregan las agregaciones cristalinas para la formación de estructuras.

Para concluir, se puede decir que se aplicaron herramientas que permiten obtener resultados más cercanos a la realidad, de este modo comparar la simulación y las agregaciones cristalinas para de esta manera ver los procesos en el fenómeno de cristalizar. Posteriormente, se realizó una indagación sobre el estado de la cuestión cuyos resultados son usados para generar diseños que permitan experimentar de tal manera que se modelizan estructuras con la capacidad de agregarse similarmente a lo observado en la experimentación y la simulación.

5.4 Proyección de Datos

5.4.1 Introducción. Basado en los datos obtenidos de la experimentación y mapeos de agregaciones cristalinas realizados durante desarrollo del *workshop* en el primer semestre del 2019, obtenida se pretende generar proyecciones que se asemejen a la realidad de la agregación en lapsos de tiempo y espacio más extensos a lo experimentado. Cabe recordar que durante el desarrollo del workshop se pidió a los participantes que generaran proyecciones en métodos libres con cada uno de sus resultados cuyos resultados al ser visualizados fueron muy lejanos de la realidad debido al uso de proyecciones lineales simples.

El grupo conformado por Miguel Ángel Quintero y Juan José Conde presentaron la proyección de datos basados en métodos de análisis de datos en código *Python*, cuyos

resultados proyectados fueron altamente cercanos a la realidad.

Una vez culminado el semestre y el *workshop* cada grupo aporto la información recolectada, incluyendo el código *Python* al *d_lab* con el fin de profundizar y continuar la investigación de la agregación de materiales cristalinos. Entonces, esta primera etapa buscó proyectar la información recolectada usando el código escrito en *Python + anaconda* con el fin de obtener datos que representen la agregación de partículas partiendo de unos datos establecidos.

Lamentablemente, no hay un previo conocimiento sobre el funcionamiento del código con la capacidad de analizar y proyectar datos a través del uso de bibliotecas de inteligencia artificial. Por lo tanto, se debe realizar un estudio de este código para de esta manera entender la función que cumple cada una de sus partes. Para posteriormente, poder generar nuevos datos de manera más eficiente ya que se tiene un control sobre el funcionamiento y las bibliotecas usadas por este código *Python*.

5.4.2 Desarrollo.

Primera semana

Una vez inició el desarrollo de las pasantías se realizó la asignación de trabajo y la familiarización con los entornos donde se realizaron las actividades.

Asignación de área y equipos de trabajo, indicaciones de método de trabajo, asignación de horarios para el cumplimiento de la pasantía, entrega de copia de llaves del aula SF 101.

Disposición de áreas de trabajo e instalación de software necesario (*Anaconda*), los demás programas necesarios ya los disponen los equipos (*Rhinoceros + grasshopper*).

Familiarización con el entorno de *Anaconda* y de escritura *Python* (Jupyter).

> Anaconda en un software de distribución libre y abierto que utiliza grandes bibliotecas de datos científicos y aprendizaje automático esto usando lenguaje

> Python o R, además permite el análisis de grandes masas de datos que pueden ser con fines predictivos o cómputos científicos, anaconda posee más de 250 bibliotecas o paquetes (Wikipedia, Anaconda, 2019).

Este proyecto usó las bibliotecas de *anaconda* con fines de análisis predictivos, además el código realizado por Joseph Fabricio Vergel se encuentra escrito dentro de un entorno vinculado a Anaconda llamado *Jupyter*, donde se escribe y se visualizan parte de los resultados.

Una vez descargado *anaconda* se procedió a abrir buscándolo ya sea desde el buscador *Windows* o desde el botón inicio.

La apertura de anaconda tardó unos segundos, una vez abierto se pudo observar los entornos de escritura *Python* disponibles como ejemplo *Jupyter*.

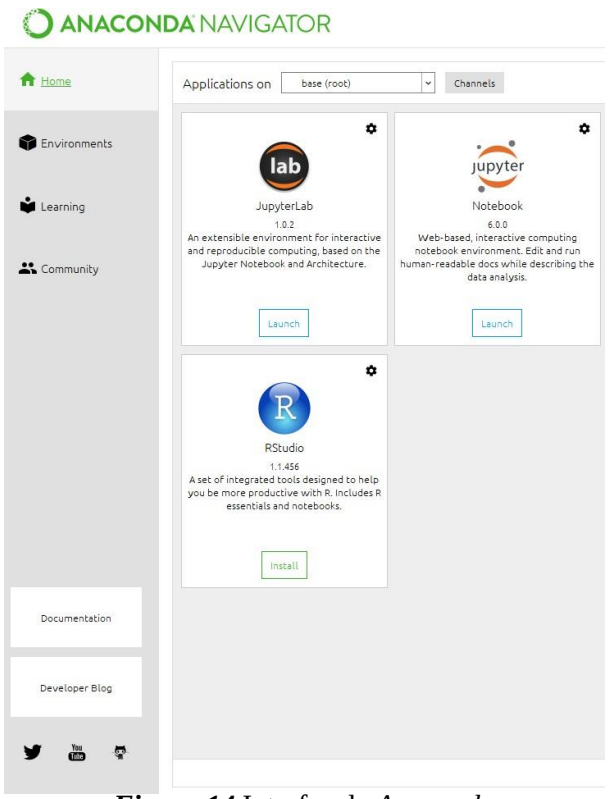


Figura 14 Interfaz de *Anaconda*

Fuente: [captura] tomado del entorno *Anaconda*

Se ubicó en el apartado de *enviroments* donde se pudo observar todas las bibliotecas disponibles por Anaconda, todas de libre uso.

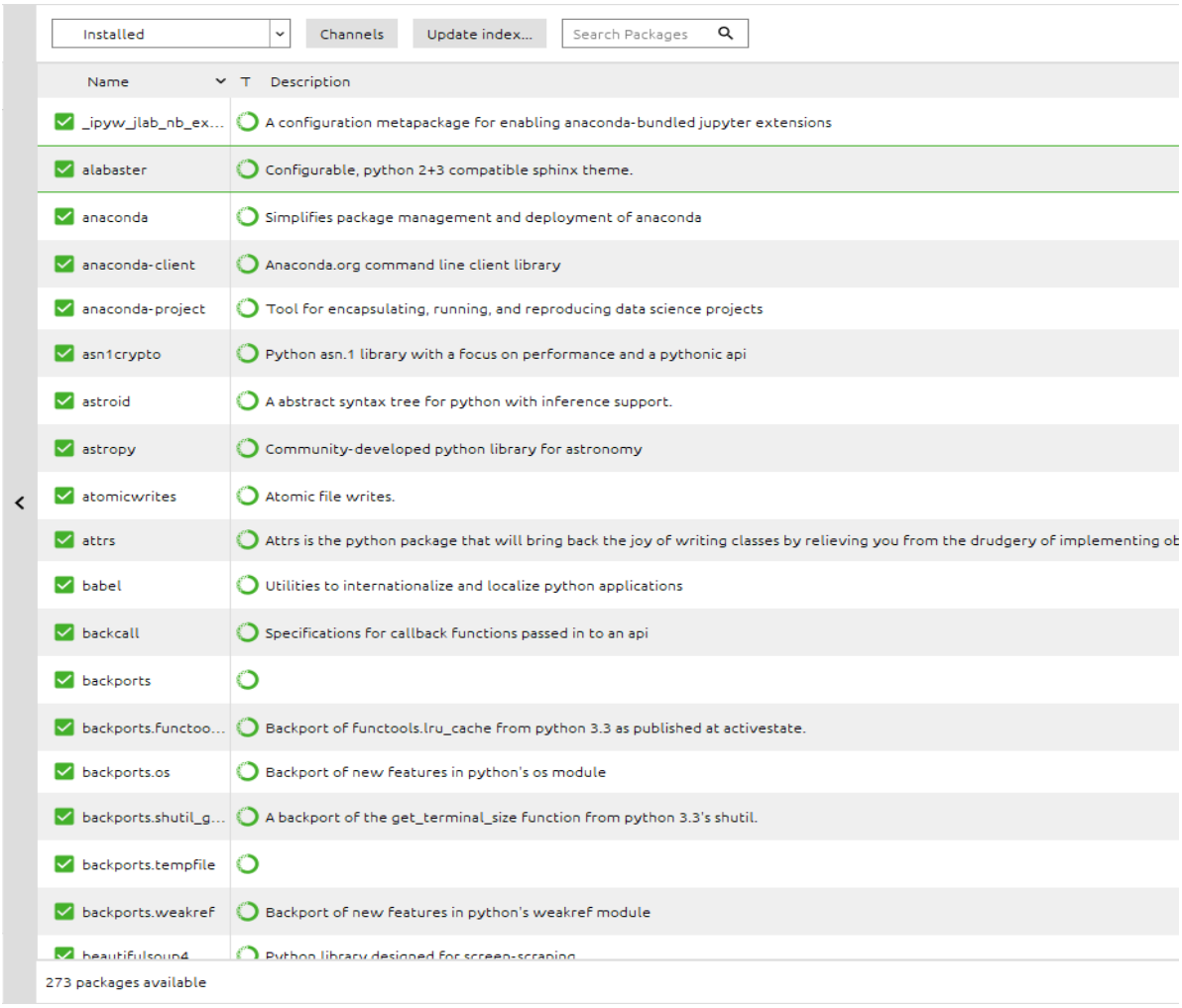


Figura 15 Bibliotecas de *Anaconda.*

Fuente: [captura] tomado del entorno *Anaconda*

En el apartado de *home*, se abrió *Jupyter notebook*, su apertura se da en *Google Chrome*.

El entorno *Jupyter* permitió escribir el código ya sea desde cero cliqueando en *new- Python* 3 o cargando un código ya escrito desde los documentos del pc.

Para este proyecto se contó con un código previamente escrito así que solo se accedió a él desde los documentos desde el ordenador.

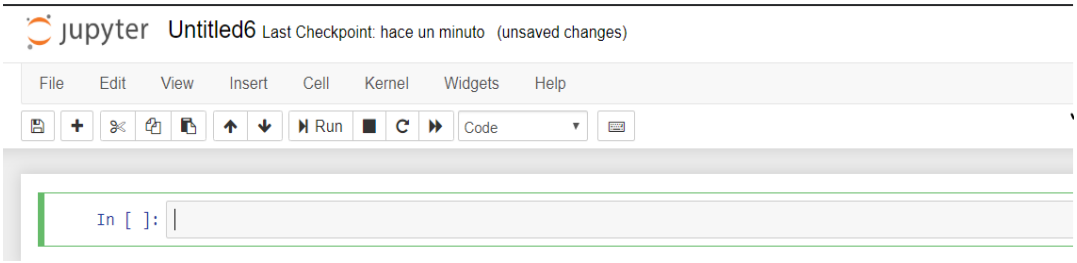


Figura 16 *Interfaz para la escritura de nuevos códigos.*
Fuente: *[captura] tomado del entorno Jupyter*

Figura 17 *Interfaz para la carga de códigos.*
Fuente: *[captura] tomado del entorno Jupyter*

Proyección de datos

Con la información obtenida durante el workshop en conjunto con el código Python se pretendió comprobar su funcionamiento, para así obtener proyecciones, que posteriormente pudieron ser visualizadas en el simulador de agregación de partículas.

Revisión del código *Python- Anaconda- Jupyter*

Para la revisión del código se debe tener en cuenta de que existen dos códigos previamente escritos que presentan el análisis para datos bidimensionales y datos tridimensionales. Para iniciar con la revisión de funcionamiento en los códigos, se usaron los datos 2d y 3d recolectados por el grupo conformado por Henry Portilla y Camila Araque. Se realizó la ejecución de *anaconda*, posteriormente la apertura de *Jupyter* y desde el explorador de archivos de este, se

ejecuta el código obtenido en el *workshop*. A cada código se le realizó una nueva copia a medida que se usen cada uno los datos recolectados en el *workshop*, pero anterior a este paso se realizará el estudio y comprobación de funcionamiento del código. Para iniciar, se importó al código las bibliotecas o paquetes de *Anaconda* necesarios para realizar el análisis y graficación de los nuevos datos a obtener, en este caso se usaron principalmente bibliotecas como *numpy*, *pandas*, *seaborn* y *matplotlib*.

```
In [44]: %matplotlib inline
         import numpy as np
         import pandas as pd
         import seaborn as sns
         from matplotlib import pyplot as plt
```

Figura 18 Importación de bibliotecas Anaconda.
Fuente: [captura] tomado del entorno Jupyter

Pandas: "En muchas situaciones del "mundo real", los datos que queremos usar proceden de múltiples archivos. Frecuentemente necesitamos combinar estos archivos en un uniquo DataFrame para analizar los datos. El paquete pandas proporciona varios métodos de combinar DataFrames incluyendo" (Martinez Paula Andrea, 2019).

Numpy: "NumPy es un paquete de Python que significa "Numerical Python", es la librería principal para la informática científica, proporciona potentes estructuras de datos, implementando matrices y matrices multidimensionales. Estas estructuras de datos garantizan cálculos eficientes con matrices" (Lidgi, 2018).

Matplotlib: "Matplotlib es probablemente el paquete de Python más utilizado para gráficos 2D. Proporciona una manera muy rápida de visualizar datos" (Rougier Nicolas, s.f.).

Seaborn: “Seaborn es una librería para Python que permite generar fácilmente elegantes gráficos. Seaborn está basada en matplotlib y proporciona una interfaz de alto nivel que es realmente sencilla de aprender” (RODRÍGUEZ, 2018)

Caso de proyección bidimensional

La proyección bidimensional hace referencia a los datos recolectados de agregaciones que surgieron en las paredes del recipiente producto de las perturbaciones del corte laser, que fueron orientados hacia el interior del recipiente y que hacen referencia al plano cartesiano para el momento de realizar los mapeos.

Insertar tablas de Excel obtenidas de los mapeos 2d

Una de las claves para importar las tablas de *Excel*, donde se encuentran los datos obtenidos de la experimentación y mapeos, se realiza usando la biblioteca de *Anaconda* llamada “pandas” expresada en el código como “pd” y usando el comando *read_excel*, permite poder leer y visualizar la tabla al momento de correr esta parte escrita.

Las siguientes imágenes corresponde al código aportado durante *workshop* el una vez reproducido presento un error, ya que su escritura presentaba una parte faltante.

```
In [86]: d2df = pd.read_excel('SEMILLAS 2D.xlsx')
         d2df.drop(['Z_0', 'Z_1', 'Z_2'], axis=1, inplace=True)
         d2df.head()
```

Out[86]:

	X_0	Y_0	X_1	Y_1	X_2	Y_2
0	10.5	16.3	10.3	8.2	9.9	4.5
1	11.0	16.3	10.6	15.7	9.6	4.4
2	10.6	15.7	11.0	7.7	9.9	4.0
3	10.5	16.0	11.5	8.0	10.1	4.1
4	11.4	16.6	11.9	8.5	10.1	4.5

Figura 19 Código entregado importación de *Excel*.
Fuente: [captura] tomado de código en *Jupyter*

```
d2df = pd.read_excel('SEMILLAS 2D.xlsx')
d2df.drop(['Z_0', 'Z_1', 'Z_2'], axis=1, inplace=True)
d2df.head()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-0b3fc71241b6> in <module>
----> 1 d2df = pd.read_excel('SEMILLAS 2D.xlsx')
      2 d2df.drop(['Z_0', 'Z_1', 'Z_2'], axis=1, inplace=True)
      3 d2df.head()

NameError: name 'pd' is not defined
```

Figura 20 Error del código
Fuente: *[captura] tomado de código en Jupyter*

El error corresponde a la falta de lectura del formato importado por parte del codigo ya que no se encuentra correctamente escrito, otro error se encuentra en la falta de informacion sobre la ubicacion de la tabla excel a exportar. Para dar solucion a este error se debió de integrar a la escritura de ubicación la letra r que hace referencia al termino “read” y posterior a esta sin dejar espacio, una comilla para luego seguir por el nombre y formato de tabla a importar y cerrar con una comilla para obtener una escritura de esta manera (r’ubicación.xlxs’) y asi la lectura de los datos sea posible.

Cabe resaltar, que existen dos métodos validos en la escritura del código para la importación de las tablas, el primero consistió en la escritura de la ubicación de las tablas dentro del ordenador donde sé está trabajando y para finalizar el formato de las tablas Excel. Se realizó de la siguiente manera: (r’C:\User\ufps\desktop\2D.xlsx’). El segundo método demandó que el archivo del código generado por *Jupyter* se encontrara en la misma carpeta con la tabla Excel de los datos obtenidos de los mapeos y de esta manera no fuera necesario la escritura de toda la ubicación, ya que el código asimilara de que el nombre y formato se encuentran dentro de la misma ubicación, se escribe de la siguiente manera: (r’2D.xlsx’).

Las tablas de datos están escritas en coordenadas x, y, z pero para este caso donde las proyecciones a realizar son bidimensionales, el eje z no se tiene en cuenta por lo tanto en el código se escribe una segunda fila (d2df. drop) en esta sección inicial para que las columnas z_0, z_1, z_2 de las tablas no sean tomadas encuentra ya que no poseen ningún tipo de información.

```
In [2]: d2df = pd.read_excel(r'C:\Users\ufps\Desktop\2D.xlsx')
        d2df.drop(['Z_0', 'Z_1', 'Z_2'], axis=1, inplace=True)
        d2df.head()
```

Out[2]:

	X_0	Y_0	X_1	Y_1	X_2	Y_2
0	3	28	3	25	2	26
1	4	26	4	25	4	28
2	6	28	4	27	4	23
3	6	28	2	24	4	27
4	1	28	2	22	3	29

Figura 21 Correcta importación de las tablas para proyección 2d.
Fuente: [captura]] tomado de código en *Jupyter*

Análisis y graficas de los datos

Una vez importadas las tablas Excel al código, se procede a generar la visualización de los datos a través de graficas que se obtiene al usar la biblioteca denominada “matplotlib” esta permite generar gráficas de datos. También, matplotlib permite ser usado en conjunto con bibliotecas científicas o estadísticas y así generar representaciones graficas de los análisis.

Para realizar los gráficos, primero se debe usar extensión plt que hace referencia a la biblioteca matplotlib, en este caso permite poder observar en gráficas los datos de la tabla, matplotlib interpreta los datos y genera gráficos, pero como se puede observar los separa según sea su organización numérica: _0, _1, _2, por lo tanto, se genera tres graficas distintas haciendo referencia a cada uno de los grupos de coordenadas.

```
plt.figure(figsize=(18,3))
plt.subplot(131)
ax = sns.scatterplot(d2df['X_0'], d2df['Y_0'],
                     color=sns.color_palette("tab10", n_colors=10)[0],
                     s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

plt.subplot(132)
ax1 = sns.scatterplot(d2df['X_1'], d2df['Y_1'],
                      color=sns.color_palette("tab10", n_colors=10)[1],
                      s=70)
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_1$')
plt.grid()

plt.subplot(133)
ax2 = sns.scatterplot(d2df['X_2'],d2df['Y_2'],
                      color=sns.color_palette("tab10", n_colors=10)[2],
                      s=70)
plt.xlabel(r'$X_2[mm]$')
plt.ylabel(r'$Y_2[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_2$')
plt.grid()
```

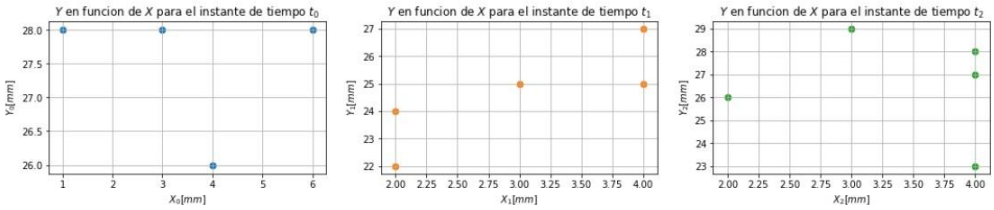


Figura 22 Correcta importación de las tablas para proyección 2d.
Fuente: *[*captura] tomado de código en *Jupyter*

Pero este tipo de gráfica que separa cada grupo de puntos no es necesaria, ya que no hay una nube de puntos integrados como sucede en la experimentación, por esto el código incluye la extensión np, que hace referencia al uso de la biblioteca Numpy que se encarga de organizar los datos en un solo grupo de columnas con el fin de observar todos los datos experimentados juntos dentro de una misma gráfica para que todos los datos sean tomados en cuenta en una misma nube de puntos al ser analizados.

```
d2df['t_0'] = np.zeros(len(d2df))
d2df['t_1'] = np.ones(len(d2df))
d2df['t_2'] = np.ones(len(d2df))*2
seed2 = np.vstack((d2df[['X_0', 'Y_0', 't_0']].values,
                   d2df[['X_1', 'Y_1', 't_1']].values,
                   d2df[['X_2', 'Y_2', 't_2']].values))
seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 't'])
seed2_df['t'] = seed2_df['t'].astype(int)
print('Dimension: ', seed2_df.shape)
seed2_df.sample(10)
```

```
Dimension:  (15, 3)
```

	X	Y	t
13	4.0	27.0	2
12	4.0	23.0	2
3	6.0	28.0	0
6	4.0	25.0	1
4	1.0	28.0	0
10	2.0	26.0	2
8	2.0	24.0	1
1	4.0	26.0	0
7	4.0	27.0	1
11	4.0	28.0	2

Figura 23 *Organización de los datos*

Fuente: *[captura] tomado de código en Jupyter*

Una vez organizados los datos en la misma tabla se procede a graficar y analizar para esto se usará dos bibliotecas como la ya mencionada *matplotlib* (plt) y *seaborn* (sns), este último permite el analizar los datos que son representados en la gráfica y además puede reconocer la ubicación de cada dato en la tabla según el grupo donde se encuentra (_0, _1, _2) y los representa en un color.

```
ax = sns.scatterplot(seed2_df['X'], seed2_df['Y'], hue=seed2_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
sns.set_palette("tab10")
plt.xlabel(r'$X[mm]$')
plt.ylabel(r'$Y[mm]$')
plt.title(r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$')
plt.grid()
```

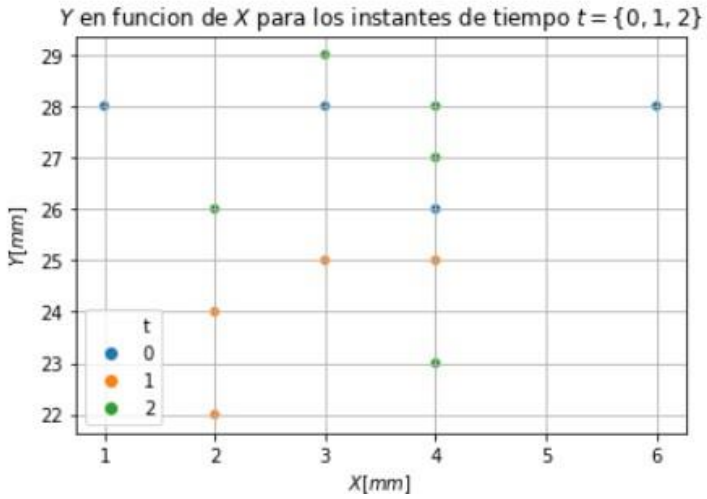


Figura 24 *Seaborn* y *matplotlib* gráficas y nube de puntos.
Fuente: [captura] tomado de código en *Jupyter*

Basado en los resultados de la gráfica anterior se procede a realizar el análisis de los datos, para iniciar se usa la biblioteca panda (pd.) para este caso se encarga de analizar las tendencias en cada uno de los planos ya sea x o y. Para el análisis *pandas* se trazaron líneas rectas entre la nube de puntos en busca de la línea más eficiente y viable para la generación de tendencias, esto lo realiza analizando individualmente cada eje y luego en conjunto, resultando así el grupo de rectas más viables entre la nube de puntos, para finalizar se usa *matplotlib* (plt.) para hace visible los resultados de este análisis.

```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=1)
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
for i in range(len(df_0)):
    plt.plot([df_0.loc[i, 'X'], df_1.loc[i, 'X']],
             [df_0.loc[i, 'Y'], df_1.loc[i, 'Y']],
             [df_1.loc[i, 'X'], df_2.loc[i, 'X']],
             [df_1.loc[i, 'Y'], df_2.loc[i, 'Y']], color=('#556270'))
plt.xlabel(r'$X[mm]$')
plt.ylabel(r'$Y[mm]$')
plt.title(r'Tendencia en $X-Y$')
plt.show()
```

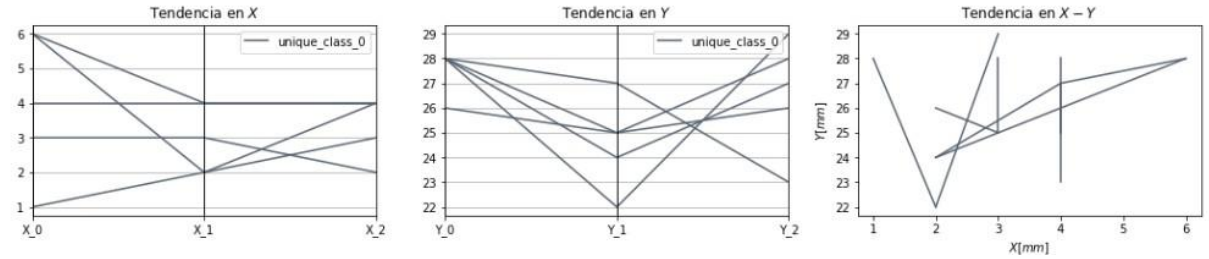


Figura 25 Recta de tendencias

Fuente: [captura] tomado de código en *Jupyter*

Una vez realizado las tendencias lineales, se procede a realizar análisis más profundos de los datos obtenidos de la experimentación, esto con fines predictivos basados en métodos eficientes para realizar esta tarea. Para esto se usa la biblioteca *seaborn* (sns.) que usa métodos avanzados para el análisis de tendencias no lineales o complejas, los métodos usados en este caso que busca la predicción de tendencias en nubes de puntos, son la regresión lineal, análisis de variables e histogramas.

```
sns.pairplot(seed2_df, hue='t', kind="reg")
plt.show()
```

```
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:48
ue_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.
n double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
```

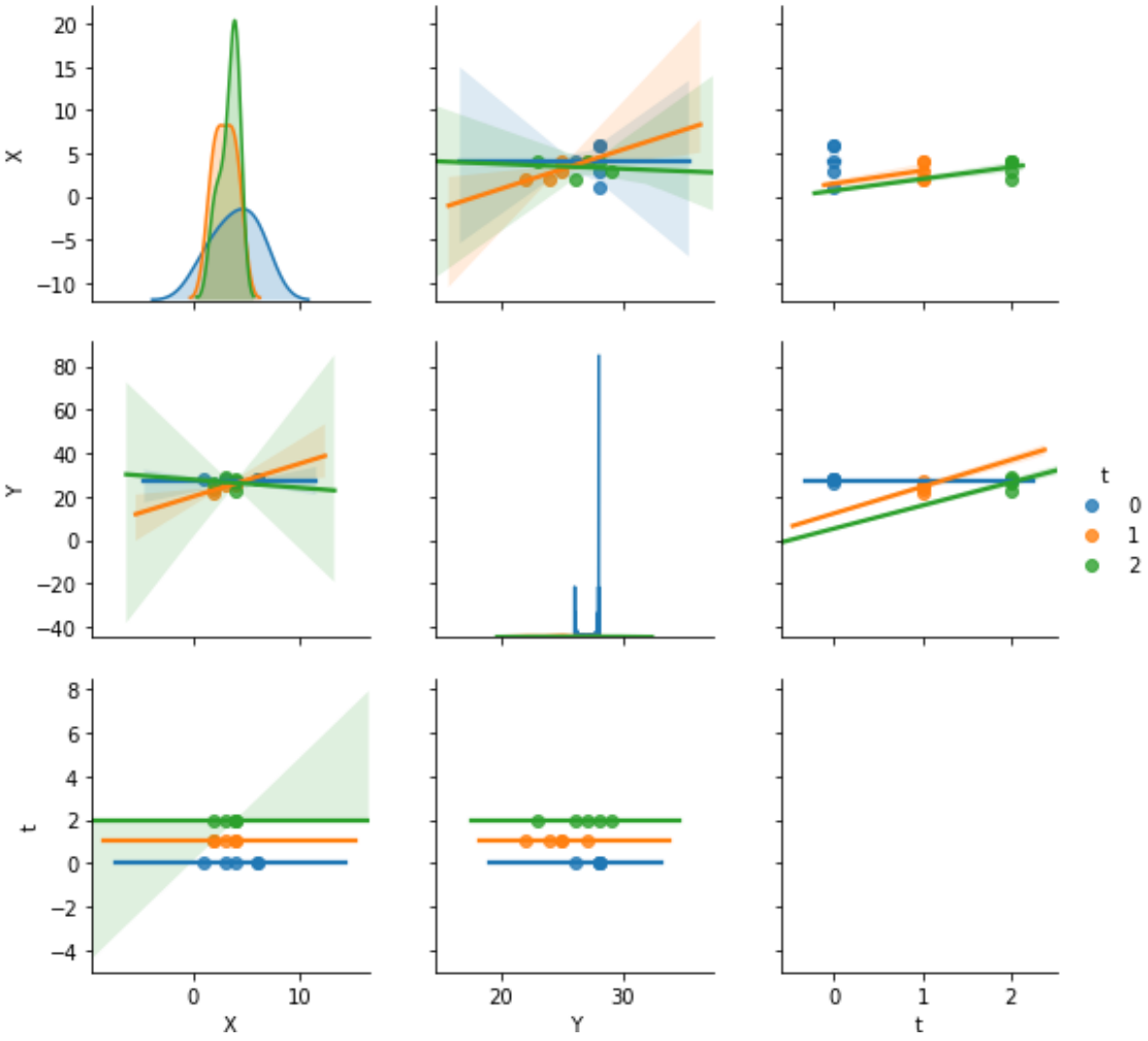


Figura 26 Análisis de variables y nube de puntos por regresión lineal.
Fuente: [captura] tomado de código en *Jupyter*

```
sns.pairplot(seed2_df, hue='t', diag_kind='hist')
plt.show()
```

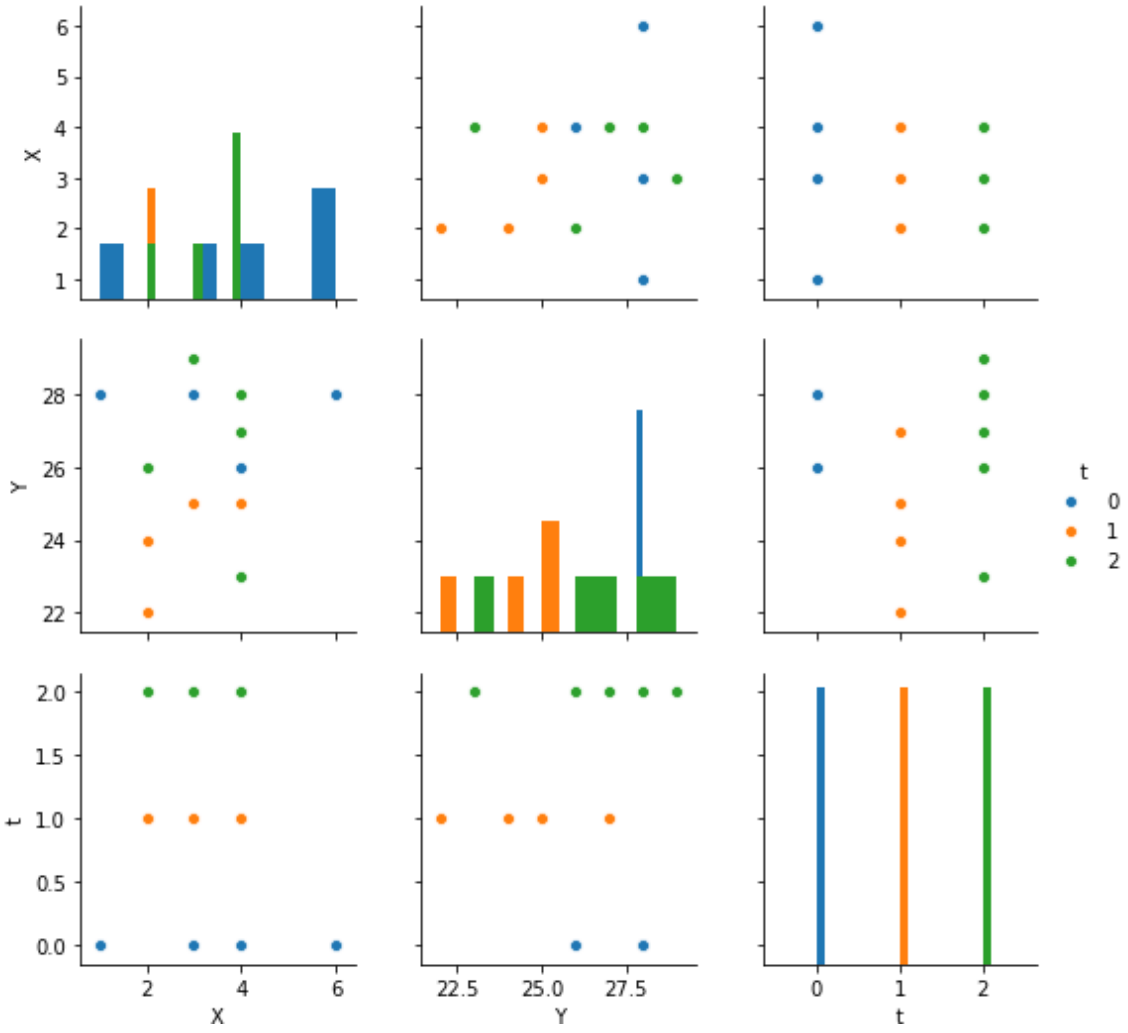


Figura 27 Análisis de histogramas

Fuente: [captura] tomado de código en *Jupyter*

```
sns.pairplot(seed2_df, hue='t')
plt.show()
```

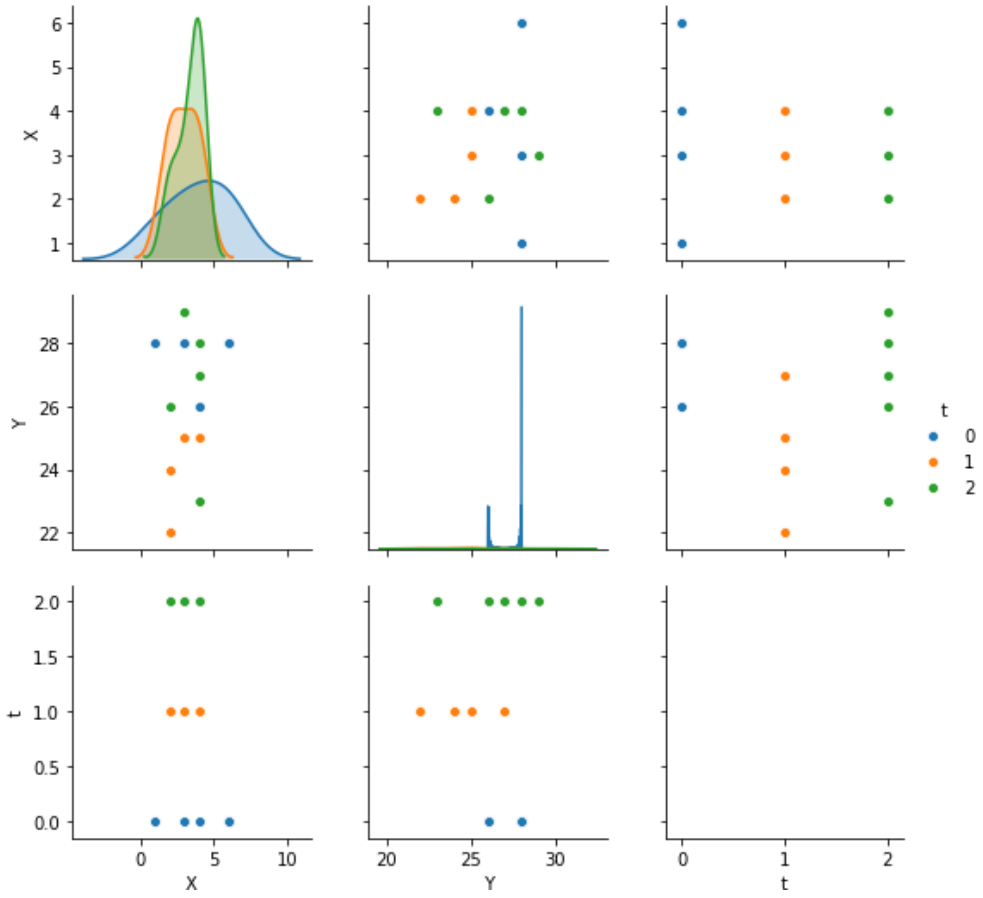


Figura 28 Análisis de variables basados en t o _0, _1, _2.
Fuente: [captura] tomado de código en *Jupyter*

Partiendo de los análisis y tendencias usados, el código posee la capacidad generar proyecciones de datos haciendo referencia al crecimiento y agregación de partículas en lapsos de tiempo y espacio más amplio a los de la realidad experimentada, pero sin alejarse de ella.

Resultado de la Proyección de Datos

Una vez obtenidos los resultados de los análisis de nubes de puntos basados en los datos obtenidos de la experimentación, se pretende tomar estos análisis para generar proyecciones por este motivo se importa una biblioteca llamada *sklearn* es la encargada de generar respuestas

basados en los análisis realizados anteriormente, esta herramienta perteneciente al aprendizaje automático usa los datos de la experimentación y los análisis para de esta manera dar resultamos más cercanos a la realidad.

Sklearn "es probablemente la librería más útil para Machine Learning en Python, es de código abierto y es reutilizable en varios contextos, fomentando el uso académico y comercial. Proporciona una gama de algoritmos de aprendizaje supervisados y no supervisados en Python" (Lidgi, 2018)

Anteriormente, dentro de la escritura perteneciente al código se usaron otras herramientas de aprendizaje automático como lo es *seaborn* o *numpy* que apoyaron y generaron análisis basados en los datos, pero *sklearn* es una biblioteca de anaconda y la principal herramienta en el código que se encarga de aprender de los análisis como las partículas se agregan para así producir proyecciones y resultados.

```
from sklearn import mixture

dpgmm = mixture.BayesianGaussianMixture(
    n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
    weight_concentration_prior_type='dirichlet_process',
    mean_precision_prior=1e-2, covariance_prior=1e0 * np.eye(2),
    init_params="kmeans", max_iter=100, random_state=2).fit(seed2_df[['X', 'Y']])
```

Figura 29 *Machine learning sklearn*.
Fuente: [captura] tomado de código en *Jupyter*

Una vez, usado el aprendizaje automático, se desea visualizar y exportar los resultados, esto se realiza usando pandas, *matplotlib*, *seaborn* y *sklearn* los cuales generan gráficos, proyectan, aprenden y organiza las proyecciones en tablas. Cabe recordar que se está estudiando el funcionamiento del código obtenido durante el desarrollo *workshop* dentro del cual no se encuentra una parte del código que es la encargada de exportar los resultados en formato texto al ordenador.

```
plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

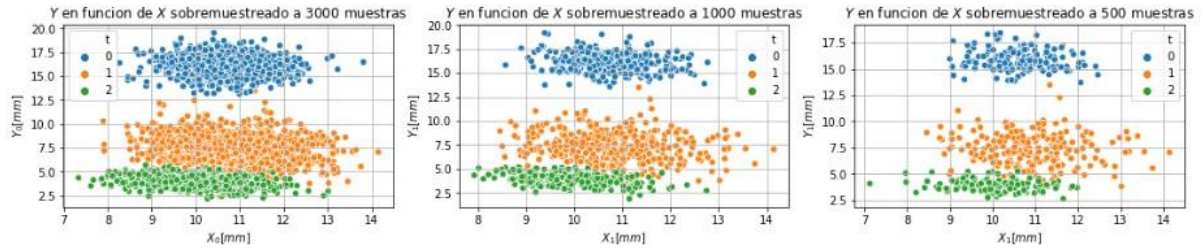


Figura 30 Resultados de las proyecciones sin exportación

Fuente: [captura] tomado de código en *Jupyter*

Para hallar la parte faltante del código se realizó una minuciosa comparación entre el código bidimensional y tridimensional dentro del apartado de exportación que es igual para los dos, en el código tridimensional se halló la parte faltante y se incluyó al bidimensional esta parte es: (seed2over_df.to_csv ('cristales_2d_3000_muestras.csv')).

Esta parte del código es de suma importancia ya que se obtienen datos proyectados que alimentaran la simulación. La exportación de los datos funciona con *sklearn* en donde se le ordena genera el numero muestras deseado en este caso su punto máximo es de 3000 agregaciones. Estos nuevos datos con el uso pandas son organizados en formato csv.

```
plt.figure(figsize=(18,3))
plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_2d_1000_muestras.csv')
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

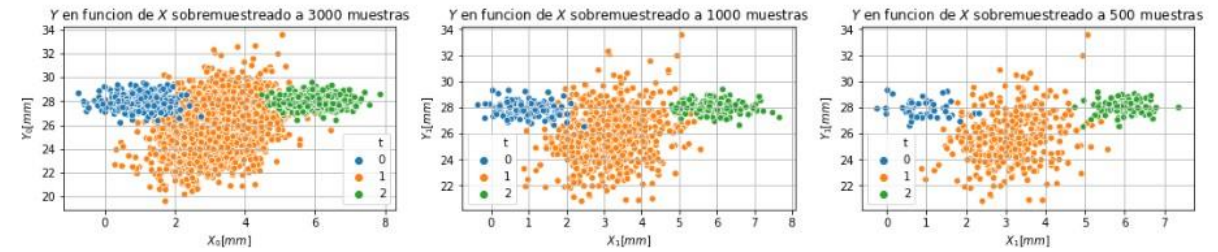


Figura 31 Resultados de las proyecciones con extensión de exportación.
Fuente: [captura] tomado de código en *Jupyter*.

Caso de Proyección Tridimensional

Durante la experimentación del *workshop* se realizaron experimentos que consisten en la agregación de cristales en filamentos suspendidos provocando de esta manera la agregación de partículas a modo multidireccional estas fueron mapeadas y organizadas en tablas con el fin de poder realizar simulaciones que permitan visualizar la dirección tridimensional y la creación de modelos en la formación de estructuras. Luego, se introdujeron los datos de la experimentación al código *Python* con el fin de generar visualizaciones y proyecciones de un posible crecimiento

y orientación sin límite de espacio, tiempo o solución, siendo estos los limitantes de la experimentación.

El código *Python* para el caso tridimensional en su estructura es muy similar a la bidimensional, aunque en la importación de tablas, los gráficos y la generación de resultados, se tuvo en cuenta los ejes X, Y, Z, caso contrario de las proyecciones bidimensionales. Para comprobar el funcionamiento del código tridimensional se usaron los datos recolectados por Camila Araque y Henry Portilla durante el desarrollo del *workshop*. Y en la importación de las tablas de la experimentación se utilizó la biblioteca *pandas*, pero en este caso es muy importante tuvo en cuenta los ejes X, Y, Z.

```
In [8]: import pandas as pd
        d2df = pd.read_excel('3D1.xlsx')
        d2df.replace(0, np.nan, inplace=True)
        d2df = DataFrameImputer().fit_transform(d2df)
        d2df
```

Out[8]:

	X_0	Y_0	Z_0	X_1	Y_1	Z_1	X_2	Y_2	Z_2
0	26	27	28	33.978022	22.461538	22.241758	40.978022	19.000000	17.000000
1	26	24	26	33.978022	22.000000	22.000000	41.956044	17.461538	16.241758
2	29	24	26	33.978022	19.000000	20.000000	42.000000	17.000000	16.000000
3	30	22	25	36.978022	19.000000	20.000000	42.000000	14.000000	14.000000
4	30	22	25	37.978022	17.000000	19.000000	45.000000	14.000000	14.000000
5	31	22	24	37.978022	17.000000	19.000000	46.000000	12.000000	13.000000
6	31	22	24	38.978022	17.000000	18.000000	46.000000	12.000000	13.000000
7	31	22	24	38.978022	17.000000	18.000000	47.000000	12.000000	12.000000
8	32	22	25	38.978022	17.000000	18.000000	47.000000	12.000000	12.000000
9	32	21	23	39.978022	17.000000	19.000000	47.000000	12.000000	12.000000
10	32	23	23	39.978022	16.000000	17.000000	48.000000	12.000000	13.000000
11	32	24	24	39.978022	18.000000	17.000000	48.000000	11.000000	11.000000
12	32	24	23	39.978022	19.000000	18.000000	48.000000	13.000000	11.000000
13	33	24	23	39.978022	19.000000	17.000000	48.000000	14.000000	12.000000

Figura 32 Importación de tabla de datos.
Fuente: [captura] tomado de código en *Jupyter*.

Posterior a la importación de las tablas se procedió a realizar los análisis de las agregaciones, aunque en este caso se debe tener en cuenta de que las gráficas deben ser tridimensionales. El código encargado de generar graficas 3d es *mplot3d* con importación de *axes3d* esto permite la generación de gráficas, para de esta manera generar análisis y respuestas en los tres ejes. Sin embargo, ejecutado esta parte del código se halla de que este no contiene la biblioteca que

permite generar graficas o análisis tridimensionales de esta manera es imposible que el resto del código llegue a funcionaren su totalidad partiendo de esta parte.

```
fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
ax.scatter(d2df['X_0'], d2df['Y_0'], d2df['Z_0'],
           color=sns.color_palette("tab10", n_colors=10)[0],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
ax.scatter(d2df['X_1'], d2df['Y_1'], d2df['Z_1'],
           color=sns.color_palette("tab10", n_colors=10)[1],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
ax.scatter(d2df['X_2'], d2df['Y_2'], d2df['Z_2'],
           color=sns.color_palette("tab10", n_colors=10)[2],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()
plt.tight_layout()
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~\Anaconda3\lib\site-packages\matplotlib\projections\__init__.py in get_projection_class(projection)
     57     try:
---> 58         return projection_registry.get_projection_class(projection)
     59     except KeyError:

~\Anaconda3\lib\site-packages\matplotlib\projections\__init__.py in get_projection_class(self, name)
     24         """
---> 25         return self._all_projection_types[name]
     26

KeyError: '3d'

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-7-cae741557e0f> in <module>
      1 fig = plt.figure(figsize=(18,3))
----> 2 ax = fig.add_subplot(131, projection='3d')
      3 ax.scatter(d2df['X_0'], d2df['Y_0'], d2df['Z_0'],
      4            color=sns.color_palette("tab10", n_colors=10)[0],
      5            s=70)
```

Figura 33 Error de generación de graficas 3d.
Fuente: [captura] tomado de código en *Jupyter*.

Para la generación de graficas tridimensionales en el código, se usa las bibliotecas anteriormente mencionadas como lo es *mplot3d* que permite las visualizaciones de nubes de puntos en espacios y coordenadas tridimensionales. Para insertar esta biblioteca se usó el comando de importación en conjunto del paquete de datos (*axes3d*) así de esta manera se pueden observar los gráficos tridimensionales.

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
ax.scatter(d2df['X_0'], d2df['Y_0'], d2df['Z_0'],
           color=sns.color_palette("tab10", n_colors=10)[0],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
ax.scatter(d2df['X_1'], d2df['Y_1'], d2df['Z_1'],
           color=sns.color_palette("tab10", n_colors=10)[1],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
ax.scatter(d2df['X_2'], d2df['Y_2'], d2df['Z_2'],
           color=sns.color_palette("tab10", n_colors=10)[2],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()
plt.tight_layout()
plt.show()
```

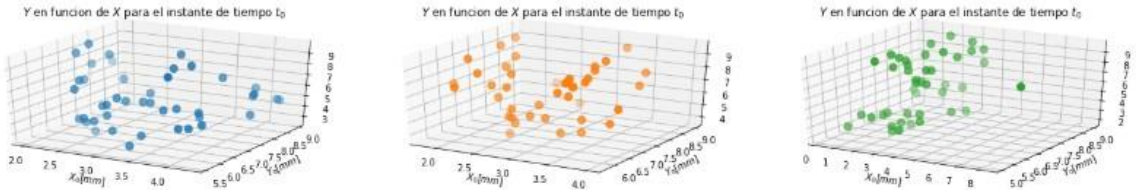


Figura 34 Biblioteca y funcionamiento del código.
Fuente: [captura] tomado de código en *Jupyter*.

Las herramientas de análisis que usa para el estudio de nubes de puntos es *sklearn* y *seaborn* las mismas que fueron usadas en la proyección bidimensional, a diferencia que en este caso usa la regresión lineal compuesta y los análisis se realizan desde los tres ejes X, Y, Z.

```
sns.pairplot(seed2_df, hue='t', diag_kind='hist')
plt.show()
```

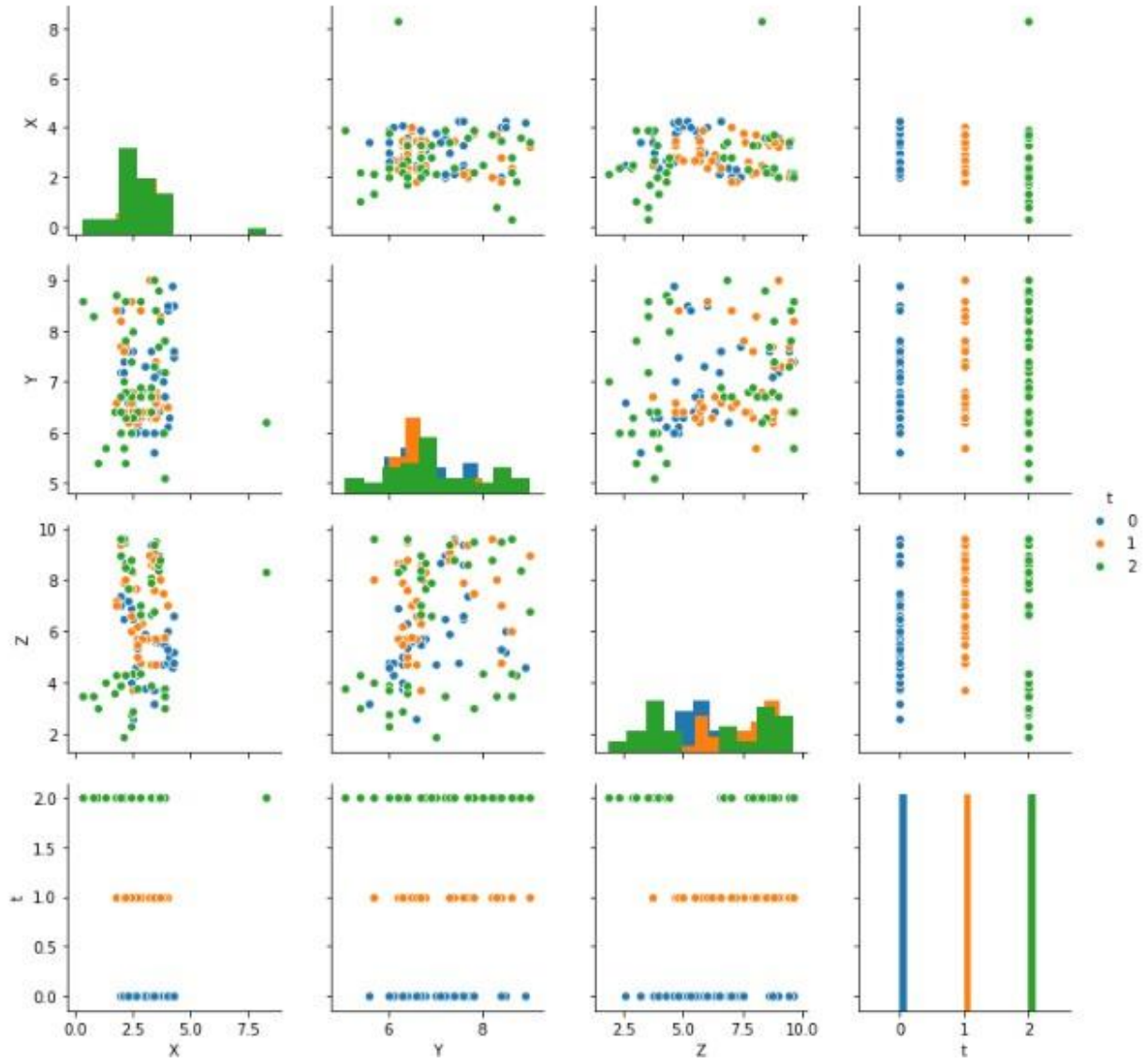


Figura 35 Análisis de datos teniendo en cuenta los ejes X, Y, Z.
Fuente: [captura] tomado de código en *Jupyter*.

Una vez realizado el análisis de la nube puntos se procedió al uso de la herramienta de *machine learning* cuya función es aprender de los análisis para la generación de proyecciones según se indique en la escritura del código, cuyos resultados son obtenidos en formato cvs.

```
ax = fig.add_subplot(132, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

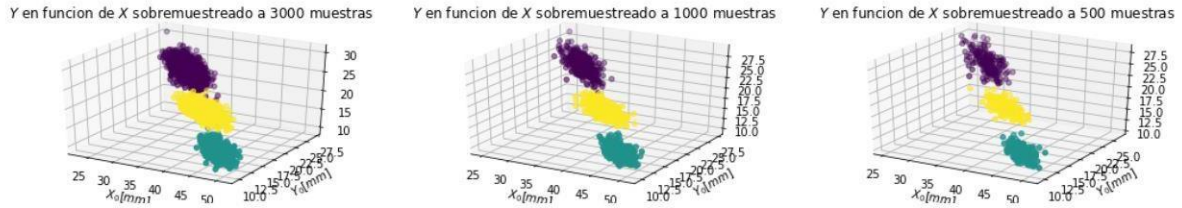


Figura 36 Resultado de proyección tridimensional.
Fuente: [captura] tomado de código en *Jupyter*.

Una vez obtenidos los resultados se aprobó el funcionamiento del código y se procedió a realizar la proyección con cada uno de los diferentes datos tridimensionales y bidimensionales captados por los participantes del *workshop*.

Posteriormente, se presentaron los resultados de las proyecciones de datos basados en la agregación de partículas ya sean bidimensionales o tridimensionales. Para observar el proceso como importación de tablas, primeros análisis, análisis de nube de puntos y resultados consultar anexos. Cabe recordar que los resultados se basaron inicialmente en los datos recolectados por cada grupo participante del *workshop*.

Resultados de Proyecciones Bidimensional

Análisis de datos y proyección completo consultar anexos B.

```
In [42]: plt.figure(figsize=(18,3))
plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('Jorge_2d_3000_muestras.csv')
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

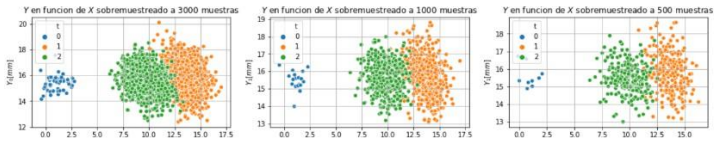


Figura 37 Resultados de las proyecciones basados en los datos de Eduardo Sandoval y Alejandra Tiria
Fuente: [captura] tomado de código en *Jupyter*.

Análisis de datos y proyección completo consultar anexos C.

```
In [17]: plt.figure(figsize=(18,3))
plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('JHON_2d_3000_muestras.csv')
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

Figura 38 Resultados de las proyecciones basados en los datos de Jhon Mantilla y Sebastián Ortega.
Fuente: *[captura] tomado de código en Jupyter*.

Análisis de datos y proyección completo consultar anexos D.

```
In [21]: plt.figure(figsize=(18,3))
         plt.subplot(131)
         X_s, t_s = dpgmm.sample(n_samples=3000)
         seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                                   pd.DataFrame(t_s, columns=['t'])], axis=1)
         seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
         seed2over_df.to_csv('Jorge_2d_3000_muestras.csv')
         ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                              hue=seed2over_df['t'],
                              palette=sns.color_palette("tab10", n_colors=3))
         plt.xlabel(r'$X_0[mm]$')
         plt.ylabel(r'$Y_0[mm]$')
         plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
         plt.grid()

         plt.subplot(132)
         X_s, t_s = dpgmm.sample(n_samples=1000)
         seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                                   pd.DataFrame(t_s, columns=['t'])], axis=1)
         seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
         ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                              hue=seed2over_df['t'],
                              palette=sns.color_palette("tab10", n_colors=3))
         plt.xlabel(r'$X_1[mm]$')
         plt.ylabel(r'$Y_1[mm]$')
         plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
         plt.grid()

         plt.subplot(133)
         X_s, t_s = dpgmm.sample(n_samples=500)
         seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                                   pd.DataFrame(t_s, columns=['t'])], axis=1)
         seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
         ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                              hue=seed2over_df['t'],
                              palette=sns.color_palette("tab10", n_colors=3))
         plt.xlabel(r'$X_1[mm]$')
         plt.ylabel(r'$Y_1[mm]$')
         plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
         plt.grid()
```

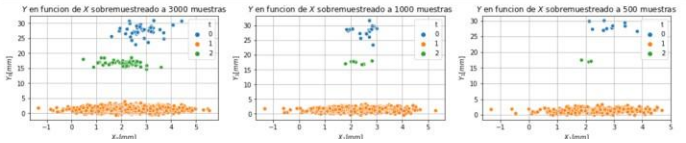


Figura 39 Resultados de las proyecciones basados en los datos de Jorge Hernández y Camilo Ramones.
Fuente: [captura] tomado de código en *Jupyter*.

Análisis de datos y proyección completo consultar anexos E.

```
plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('CRISTINA_2d_3000_muestras.csv')
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
ax = sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],
                     hue=seed2over_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

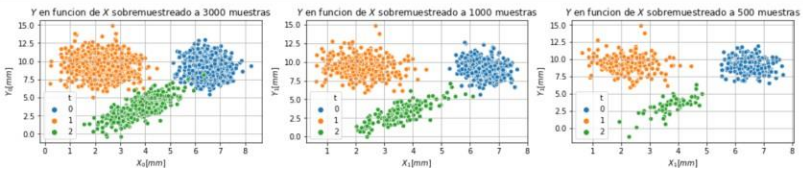


Figura 40 Resultados de las proyecciones basados en los datos de Jordi Suarez y Cristina Lizcano.
Fuente: [captura] tomado de código en *Jupyter*

Resultados de proyecciones tridimensional

Análisis de datos y proyección completo consultar anexos F.

```
In [18]: fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

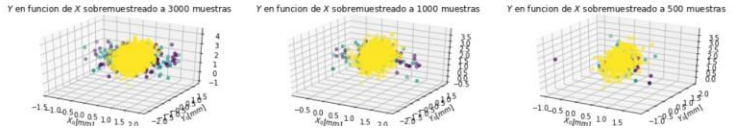


Figura 41 Resultados de las proyecciones 3d basados en los datos de Eduardo Sandoval y Alejandra Tiria.

Fuente: [captura] tomado de código en *Jupyter*.

Análisis de datos y proyección completo consultar anexos G.

```
In [17]: fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

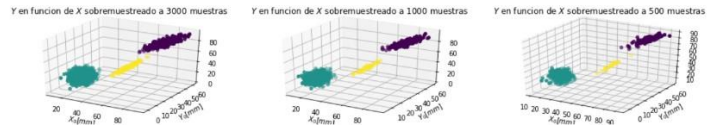


Figura 42 Resultados de las proyecciones 3d basados en los datos de Jhon Mantilla y Sebastián Ortega.

Fuente: [captura] tomado de código en *Jupyter*.

Análisis de datos y proyección completo consultar anexos H.

```
In [23]: fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

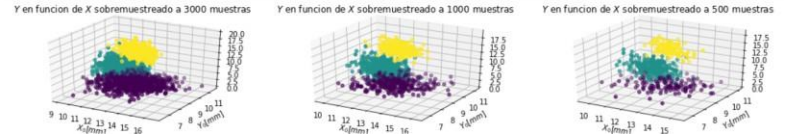


Figura 43 Resultados de las proyecciones 3d basados en los datos de Jorge Hernández y Camilo Ramones.

Fuente: [captura] tomado de código en Jupyter.

Análisis de datos y proyección completo consultar anexos I.

```
In [18]: fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
                          pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],
           c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras')
plt.grid()
```

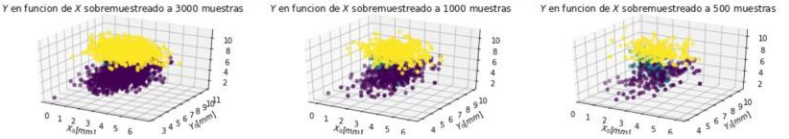


Figura 44 Resultados de las proyecciones 3d basados en los datos de Jordin Suarez y Cristina Lizcano.

Fuente: [captura] tomado de código en *Jupyter*.

5.4.3 Conclusión. Como se mencionó anteriormente, la generación de estas proyecciones permitió predecir las posibilidades de agregación de partículas cristalinas, esto con el fin de poder aprender de la materia cristalina para orientarla o generar modelos de estructuras basado en la observación y comparación entre la realidad y la simulación.

El uso de *anaconda* y *Python* no solo permitió de abrirse al uso de nuevas herramientas, sino que pudo generar visualizaciones multidimensionales de datos reales y a través del uso de la inteligencia artificial obtener nuevos datos que proyecta la información original, pero sin alejarse de la realidad, se tuvo en cuenta que estos nuevos datos son especulaciones de una posible realidad mas no es la realidad absoluta.

Realizar la proyección bidimensional y tridimensional de los datos permitió que estas nuevas coordenadas, que representan agregaciones, pudieran ser llevadas y visualizadas en la simulación para comprobar las posibilidades de orientación y formación de estructuras relevantes. Así mismo, al obtener los nuevos datos proyectados no es suficiente para el análisis de agregaciones, por lo tanto, se requiere poder visualizar dentro de un entorno la agregación de las partículas para observar se formación en el tiempo y el espacio, por esto es necesario insertar los nuevos datos en una simulación ya que además la proyección está representada en datos de partículas que se agregan más no existe una visualización en el espacio que permita una mejor comparación.

5.5 Simulación de Datos Proyectados

5.5.1 Introducción. La simulación permite observar una realidad y la posibilidad de dirección sobre las agregaciones de materia cristalina y de esto obtener modelos bidimensionales o tridimensionales que pudieron ser comparados con la realidad, los resultados pueden ser usados

como bases para la modelización de estructuras relevantes. Inicialmente, en esta etapa se realizará el estudio de funcionamiento del simulador, esto con el fin de entender la función cada uno de los nodos que componen la definición diseñada en *grasshopper*, que es capaz de interpretar los datos proyectados y los expone como agregaciones en el espacio.

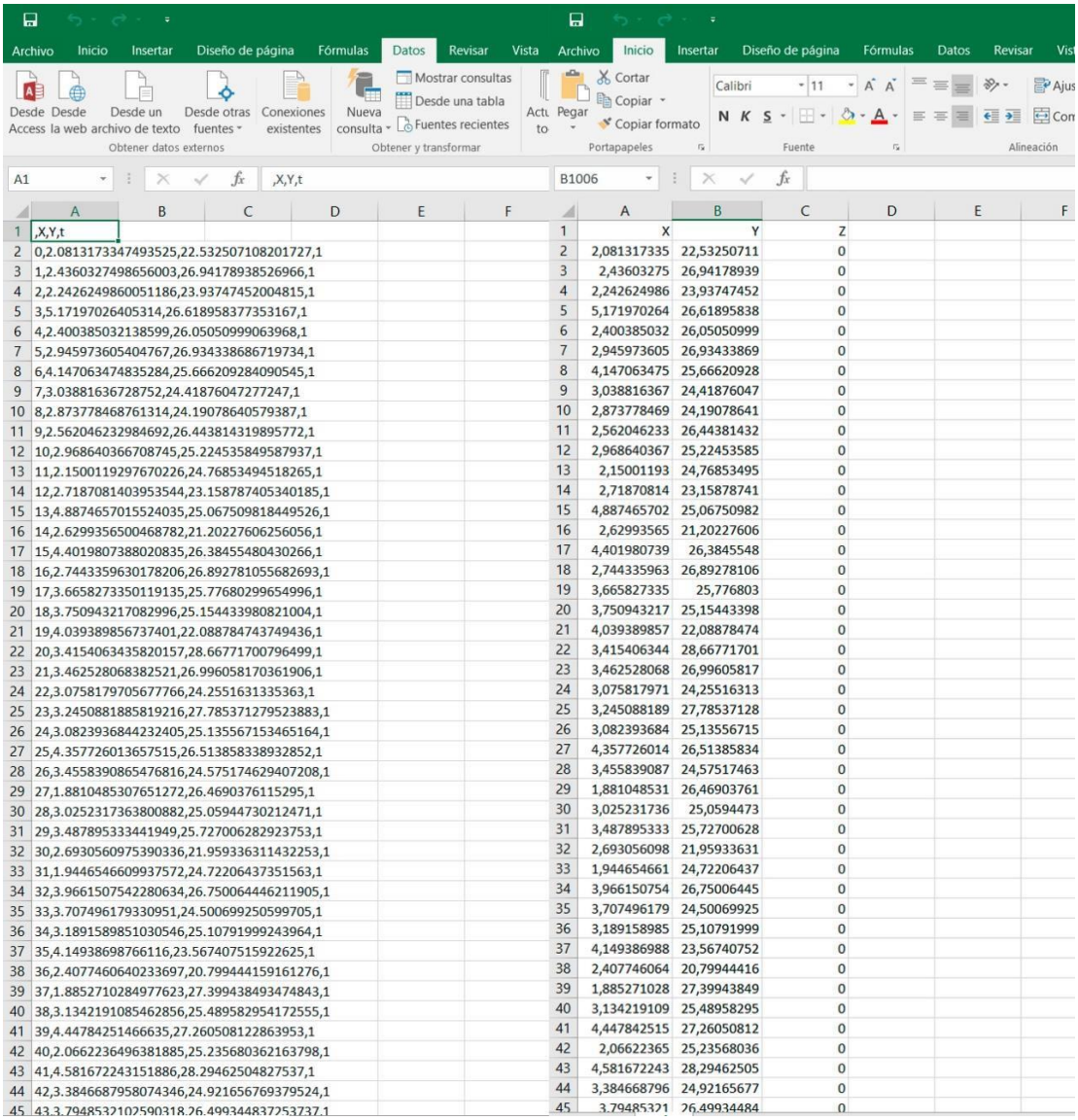
Una vez el funcionamiento de la definición fue estudiado se tomará cada uno de los datos obtenidos de la proyección y serán simulados para visualizar la agregación de partículas en el tiempo y el espacio, para este caso la simulación más que herramienta para visualizar la posibilidad de orientación de agregación basada en los datos obtenidos de la proyección, funciona también como herramienta de comparación entre la realidad y la simulación.

De los modelos obtenidos a simulación se pueden sentar bases para la creación modelos. Todo esto parte del estudio de los fenómenos presentes en la cristalización de los materiales y el análisis de la adición de partículas formadas en la simulación, adicionalmente, también se realizó la investigación de referentes que han estudiado la agregación de partículas y la creación de modelos basados en esto.

5.5.2 Desarrollo.

Simulación de Datos Proyectados

En base a los resultados obtenidos en las proyecciones, se procede a organizar esta información que se encuentra en formato texto a tablas Excel, para de esta manera traducir los datos a puntos en el espacio y así visualizar las agregaciones en la simulación desarrollada durante el *workshop*.



	A
1	,X,Y,t
2	0,2.0813173347493525,22.532507108201727,1
3	1,2.4360327498656003,26.94178938526966,1
4	2,2.2426249860051186,23.93747452004815,1
5	3,5.17197026405314,26.618958377353167,1
6	4,2.400385032138599,26.05050999063968,1
7	5,2.945973605404767,26.934338686719734,1
8	6,4.147063474835284,25.666209284090545,1
9	7,3.03881636728752,24.41876047277247,1
10	8,2.873778468761314,24.19078640579387,1
11	9,2.562046232984692,26.443814319895772,1
12	10,2.968640366708745,25.224535849587937,1
13	11,2.1500119297670226,24.76853494518265,1
14	12,2.7187081403953544,23.158787405340185,1
15	13,4.8874657015524035,25.067509818449526,1
16	14,2.6299356500468782,21.20227606256056,1
17	15,4.4019807388020835,26.38455480430266,1
18	16,2.7443359630178206,26.892781055682693,1
19	17,3.6658273350119135,25.77680299654996,1
20	18,3.750943217082996,25.154433980821004,1
21	19,4.039389856737401,22.088784743749436,1
22	20,3.4154063435820157,28.66771700796499,1
23	21,3.462528068382521,26.996058170361906,1
24	22,3.0758179705677766,24.2551631335363,1
25	23,3.2450881885819216,27.785371279523883,1
26	24,3.0823936844232405,25.135567153465164,1
27	25,4.357726013657515,26.513858338932852,1
28	26,3.4558390865476816,24.575174629407208,1
29	27,1.8810485307651272,26.4690376115295,1
30	28,3.0252317363800882,25.05944730212471,1
31	29,3.487895333441949,25.727006282923753,1
32	30,2.6930560975390336,21.959336311432253,1
33	31,1.9446546609937572,24.72206437351563,1
34	32,3.9661507542280634,26.750064446211905,1
35	33,3.707496179330951,24.500699250599705,1
36	34,3.1891589851030546,25.10791999243964,1
37	35,4.149386987661 16,23.567407515922625,1
38	36,2.4077460640233697,20.799444159161276,1
39	37,1.8852710284977623,27.399438493474843,1
40	38,3.1342191085462856,25.489582954172555,1
41	39,4.44784251466635,27.260508122863953,1
42	40,2.0662236496381885,25.235680362163798,1
43	41,4.581672243151886,28.29462504827537,1
44	42,3.3846687958074346,24.921656769379524,1
45	43,3.7948532102590318,26.499344837253737,1

	A	B	C
1	X	Y	Z
2	2,081317335	22,53250711	0
3	2,43603275	26,94178939	0
4	2,242624986	23,93747452	0
5	5,171970264	26,61895838	0
6	2,400385032	26,05050999	0
7	2,945973605	26,93433869	0
8	4,147063475	25,66620928	0
9	3,038816367	24,41876047	0
10	2,873778469	24,19078641	0
11	2,562046233	26,44381432	0
12	2,968640367	25,22453585	0
13	2,15001193	24,76853495	0
14	2,71870814	23,15878741	0
15	4,887465702	25,06750982	0
16	2,62993565	21,20227606	0
17	4,401980739	26,3845548	0
18	2,744335963	26,89278106	0
19	3,665827335	25,776803	0
20	3,750943217	25,15443398	0
21	4,039389857	22,08878474	0
22	3,415406344	28,66771701	0
23	3,462528068	26,99605817	0
24	3,075817971	24,25516313	0
25	3,245088189	27,78537128	0
26	3,082393684	25,13556715	0
27	4,357726014	26,51385834	0
28	3,455839087	24,57517463	0
29	1,881048531	26,46903761	0
30	3,025231736	25,0594473	0
31	3,487895333	25,72700628	0
32	2,693056098	21,95933631	0
33	1,944654661	24,72206437	0
34	3,966150754	26,75006445	0
35	3,707496179	24,50069925	0
36	3,189158985	25,10791999	0
37	4,149386988	23,56740752	0
38	2,407746064	20,79944416	0
39	1,885271028	27,39943849	0
40	3,134219109	25,48958295	0
41	4,447842515	27,26050812	0
42	2,06622365	25,23568036	0
43	4,581672243	28,29462505	0
44	3,384668796	24,92165677	0
45	3,79485321	26,49934484	0

Figura 45 Izquierda datos en formato texto csv. Derecha textos organizados en las tablas excel
Fuente: [captura] tomado de código en *Jupyter*.

Para este caso en el que se desea conocer el funcionamiento a fondo de la simulación desarrollada, se usarón los datos proyectados productos de la experimentación realizada por Camila Araque y Henry Urley Portilla Delgado.

Estos datos fueron introducidos a la simulación que fue desarrollada en el *workshop* por Pablo Rey, Juan Manuel Villa, Miguel Quintero y Henry Portilla, este último realizó modificaciones con el fin de reemplazar los puntos en el espacio por volúmenes. Para la construcción de la

simulación se usó la herramienta *grasshopper* un *plugin* de *Rhinoceros* que permite modelar o crear simulaciones desde nodos de algoritmos a esto se le conoce como definición. Para la creación de la definición del simulador, se inicia con el uso de la herramienta o nodo encargado de la lectura de formatos *excel* (*exdread*) a la cual se le incluyen tres nodos: El primero (*path*) se encarga de importar los archivos *Excel* a la simulación. El segundo (*toggle*) encargado de habilitar el paso de la información al nodo central (*exdread*).

Y, el tercer nodo (*bloc*) es el encargado de delimitar las columnas del archivo *Excel* para ser interpretadas por *Exread* que posteriormente serán convertidas en coordenadas en el espacio, esto quiere decir que, al escribir en el *nodo bloc* A2:A3001., se está delimitando A como la columna que se va tomar y los números (2, 3001) siendo la fila inicial y la fila final, se incluyen tres nodos bloc siendo estos un grupo.

Posterior a esto, a cada grupo *nodos bloc* se le asignaron un color, estos colores hacen referencia a las columnas usadas en este caso son A, B, C, con el fin de reconocer más fácilmente cada grupo de coordenadas ya que cada nodo hace referencia a X, Y, y si es el caso tridimensional Z.

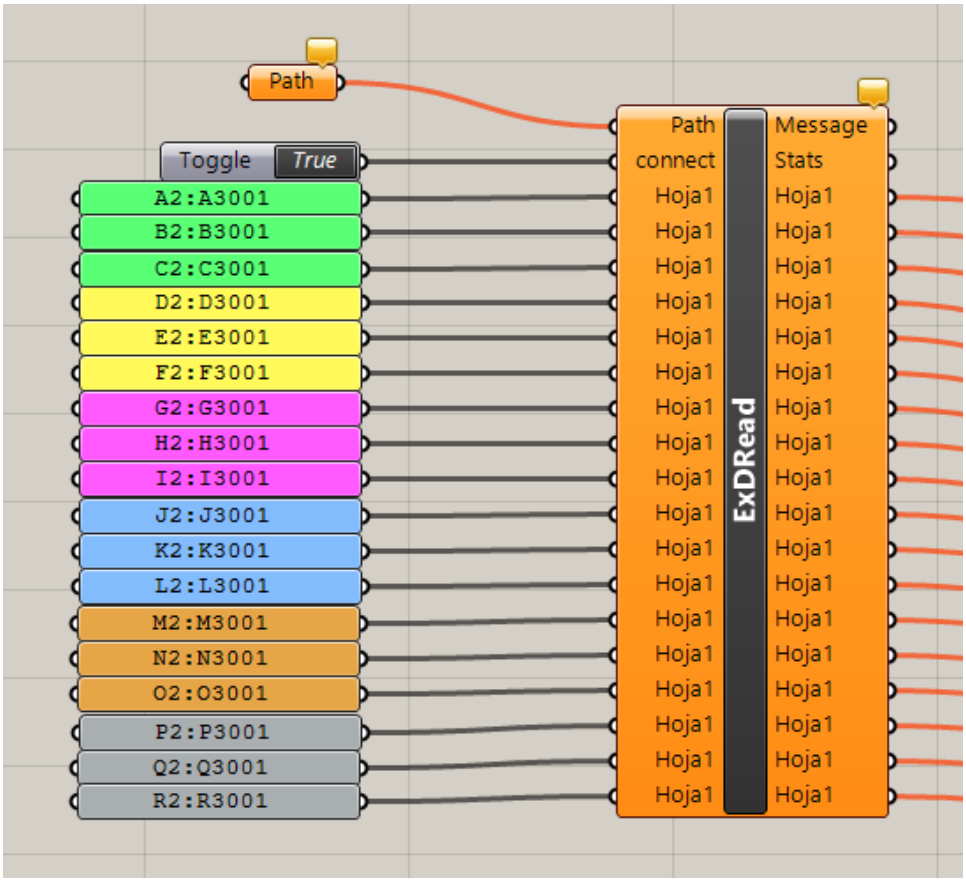


Figura 46 Importación y lectura de las tablas Excel en simulación**.**
Fuente: [captura] tomado de definición de simulador en *grasshopper*

Una vez organizada las columnas y las filas que el simulador debe interpretar, se agruparon para de esta manera traducir el conjunto de datos a coordenadas. Por este motivo, se añadieron 3 *nodos ítem list* que son conectados a *exdread* y paralelamente al grupo del *nodo bloc* que hace referencia a los datos, *ítem list* se encarga leer de manera ordenada una a una las filas asignada a *bloc*, por lo tanto, se añade el nodo *pt* o punto.

El nodo punto, posee unos conectores denominados X, Y, Z a los cuales se enlazaron cada uno de los nodos ítem list, de esta manera el nodo punto recibió la información de tres columnas de manera automática, haciendo que este nodo interpretara cada grupo de tres números como coordenadas en el espacio representado como un punto.

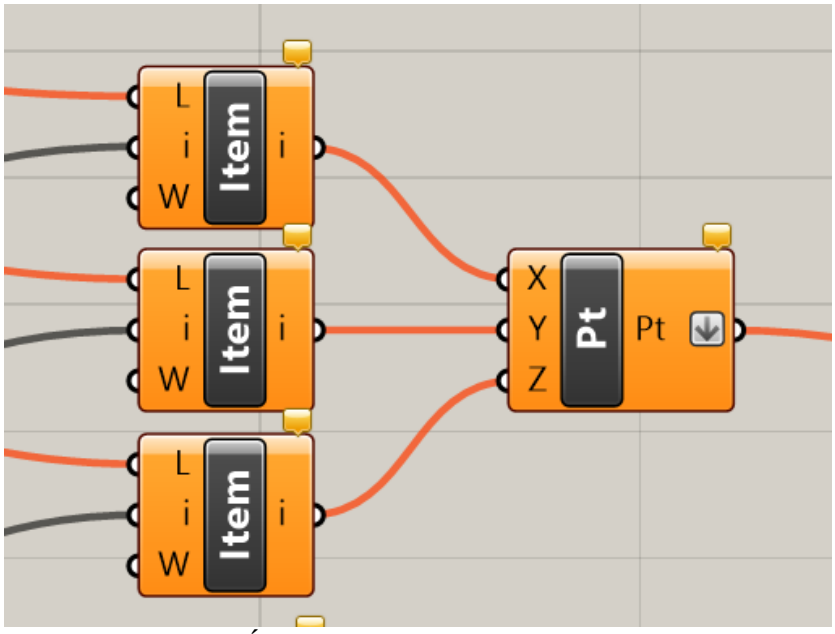


Figura 47 Ítem list conectado a el *nodo pt* o punto**.**
Fuente: [captura] tomado de definición de simulador en *grasshopper*

El nodo *restartcount*, se encarga de realizar el conteo uno a uno de todas las filas de la tabla Excel, con este nodo se pudo controlar la velocidad en el conteo de datos y los límites de lectura de estos, para el funcionamiento de este nodo se enlazó a un nodo llamado *num* y de este se desprendieron conexiones hasta el apartado *index* del nodo *ítem list*. Asimismo, el nodo *restartcount* permitió representar punto a punto cada coordenada de las tablas Excel para de este modo visualizar las agregaciones obtenidas tanto de la experimentación como los datos proyectados.

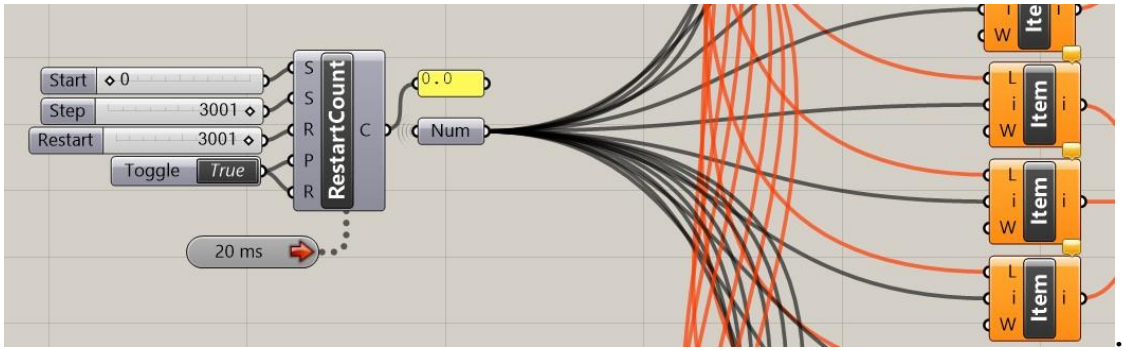


Figura 48 *Restartcount* conectado a *num* y a su vez a *ítem list.*
Fuente: *[captura] tomado de definición de simulador en grasshopper*

El nodo *restartcount*, presentó un problema al momento de simular ya que cada punto es representado uno al tiempo, esto quiere decir que cada punto se visualizó temporalmente hasta la aparición del próximo punto. Por este motivo, se incluyó el nodo *rec* que se encargó de mantener cada punto en el espacio una vez *restartcount* lo representa, de este modo se puede apreciar la aparición de las agregaciones observadas y proyectadas.

Figura 49 Nodo *rec.*
Fuente: [captura] tomado de definición de simulador en *grasshopper.*

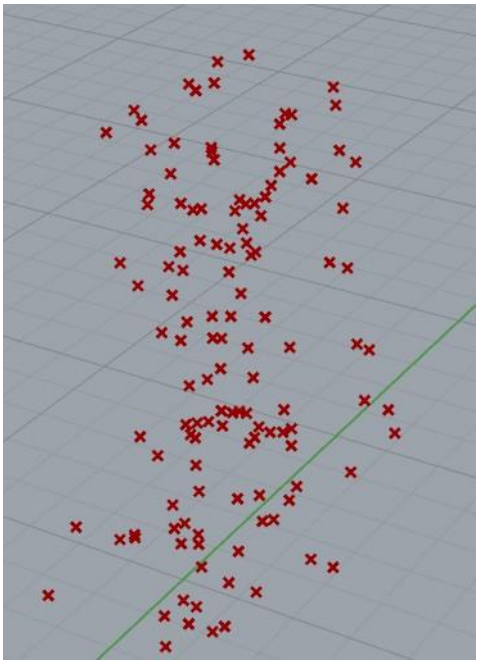


Figura 50 Nube de puntos resultante del uso del nodo *rec.*
Fuente: [captura] tomado de visualizador en *rhinoceros.*

Una vez generada la nube de puntos se pretendió reemplazar el punto por elementos

volumétricos que representen las agregaciones cristalinas de la manera más cercana a la realidad por ende se usa la representación de un cubo y además con esto se buscó poder observar la intersección entre las agregaciones observando cómo construye sus estructuras.

Por lo tanto, se usó el nodo *centerbox* cuya función fue representar cada punto como un cubo, adicionalmente se conectó al nodo *random* y a este unos *slider* que representan el rangos de dimensión de los volúmenes. Por ello la nube de puntos fue reemplazada en tiempo real por volúmenes que permitieron observar las agregaciones simuladas.

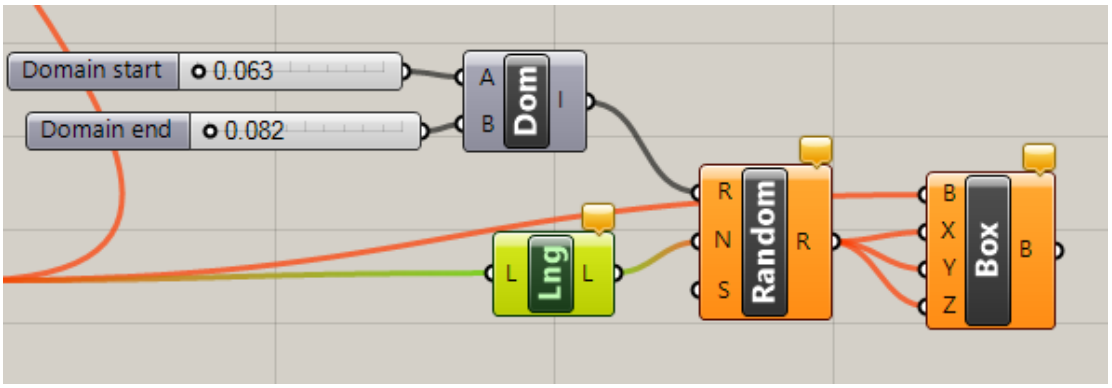


Figura 51 Generador de volúmenes cúbicos.
Fuente: [captura] tomado de definición de simulador en *grasshopper*.

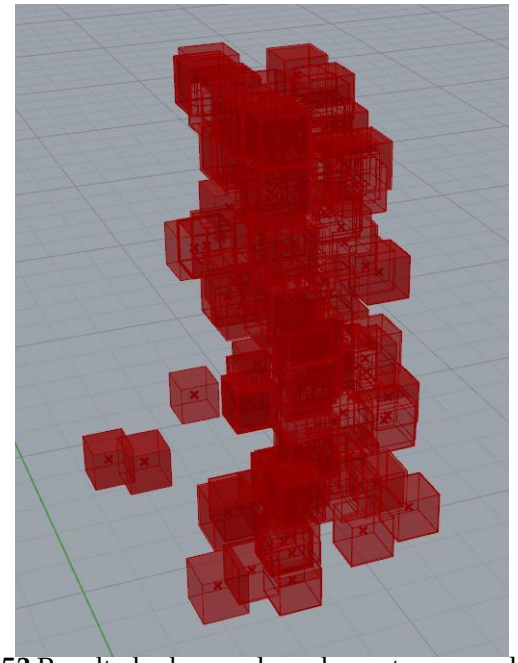


Figura 52 Resultado de remplazo de puntos por volúmenes**.**
Fuente: *[captura] tomado de visualizador en Rhinoceros.*

Una vez reproducido el código empezaron a surgir cada una de las agregaciones en forma de volúmenes cúbicos individuales debido a la proximidad entre ellos se generan intersecciones de esta empiezan a conformar estructuras aglomeradas que representan las proyecciones y las formaciones cristalinas.

Comparación de Modelos y Experimentación

Las estructuras obtenidas de la simulación ya sean las tridimensionales o bidimensionales permitieron observar las posibilidades de agregaciones si la solución tenia mayor concentración de material y mayores tiempos de cristalización. Cabe recordar, que los mapeos recolectados durante la experimentación equivalen a dos horas de cristalización tiempo que al ser simulados se representa en solo 10 segundos. Mientras que los datos proyectados poseen un total de 3000 nuevas agregaciones que al ser simulados son visualizados totalmente en un tiempo que oscila entre 2 y 5 minutos, estas agregaciones en tiempo real equivalen a una experimentación de 10 días.

Los nuevos modelos obtenidos son la posibilidad de crecimiento y agregación de partículas que parten de datos obtenidos de la experimentación de manera que estos puedan ser más próximos a la realidad. La intención de estos modelos es comparar y analizar que los modelos son similares a la realidad. Por lo tanto, para realizar la comparación se realizó un paralelo entre el modelo obtenido de la simulación proyectada, el mapeo de la experimentación y la proyección hecha en el desarrollo del *workshop,* todo esto perteneciente al miso grupo de datos. Esto se realizó, con el fin de verificar que todas las partes comparadas fueran similares entre ellas, para de esta manera comprobar si la orientación en las agregaciones es viable con respecto a lo

experimentado, y adicional, obtener información sobre como las partículas se aglomeran para de esta manera definir principios sobre la modelización de estructuras relevantes.

Comparativas de Simulaciones Bidimensionales

Los datos bidimensionales son aquellos que fueron tomados de las agregaciones crecientes en una de las paredes laterales del recipiente donde se realizó la experimentación, los cuales presentaron perturbaciones provocadas por el corte a laser. Entonces, las agregaciones crecieron por las perturbaciones, una vez estaban ocupadas totalmente, las nuevas partículas se agregaron a los lados de las partículas anteriores, provocando de esta manera que las paredes laterales del recipiente queden ocupadas por las agregaciones cristalina.

Pero la concentración de la solución cristalina en el recipiente no fue suficiente para cubrir las paredes laterales totalmente, por lo tanto, los datos que fueron recolectados y exportados al código *Python* y así se obtuvieron resultados de agregación de la posibilidad de ocupación del material cristalino. La primera comparación se trata de los datos mapeados y recolectados por el grupo conformados por Cristina Lizcano y Jordin Suarez, cuyo método de mapeos se basó en la recolección de puntos de crecimiento de cada semilla y la agregación de partículas, consecuentemente, los datos que fueron organizados en tablas correspondientes a crecimiento y las agregaciones.

La proyección realizada durante el *workshop* se basa en la réplica de patrones en el uso de tendencia lineal dando como resultado un modelo de proyecciones con un crecimiento orientado a una sola dirección en la que cada partícula se encuentra alineada, resultado muy lejano de la realidad. Sin embargo, se toman los datos obtenido de la experimentación y exportados al código

Python, ya al comprobar su funcionamiento, se importaron los datos al código donde fueron analizados y posteriormente proyectados.

Los datos obtenidos son reproducidos y visualizados en la simulación donde se obtuvo la aparición de agregaciones que ocuparon o rellenaron el espacio en tres principales aglomeraciones. Al comparar el modelo proyectado en el *workshop* con el proyectado, se pudo definir que es el más cercano a la posibilidad de agregación de partículas, a pesar que el mapeo que realizó este grupo no solo se basó en las agregaciones sino en el crecimiento también.

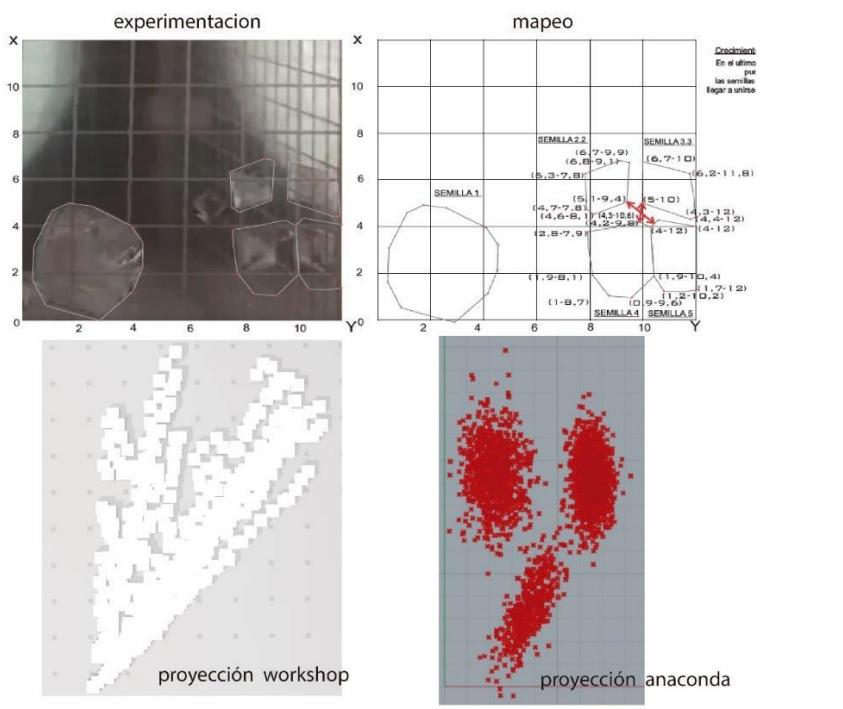


Figura 53 Agregación de partículas

Nota: Arriba mapeos 2d (elaborado, Cristina Lizcano, Jordin Suarez), abajo izquierda simulación 2d de proyección workshop (elaborado, Cristina Lizcano, Jordin Suarez), derecha simulación 2d de proyección (Henry Urley Portilla)**.** [Cuadro comparativo]

La toma de datos de los demás grupos participantes en el *workshop* realizó los mapeos basados solo en la agregación de las partículas agregadas y no de puntos de crecimiento, lo que permitió visualizar al momento de la simulación, una comprensión sobre la formación de

estructuras a partir de las agregaciones y entender a través de la observación de cómo crecen y se forman las estructuras agregadas.

Por lo tanto, al someter los grupos de datos a la proyección con el código *Python*, en donde son ordenados, analizados y proyectados se obtienen nuevos datos de 3000 muestras nuevas. Al visualizar los nuevos datos en la simulación se puede observar como el fenómeno de agregación de las partículas avanza más con respecto a lo observado en la experimentación o cercano a la realidad.

Esto, consiste en que las agregaciones se concentran inicialmente en los puntos o coordenadas que hacen referencia a las perturbaciones por corte a laser, esto es lo observado incluso en la experimentación, siguiente a esto se observa como las agregaciones ocupan los espacios vacíos hasta conformar una superficie aparentemente compacta, que nace de la creación de partículas individuales.

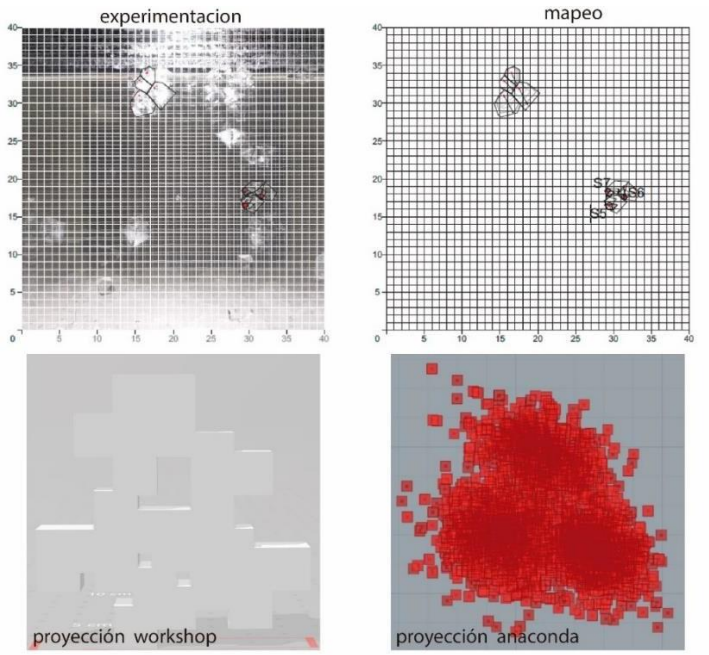


Figura 54 Agregación de partículas

Nota: Arriba mapeos 2d (elaborado, Jhon Mantilla, Sebastián Ortega), abajo izquierda simulación 2d de proyección *workshop* (elaborado, Jhon Mantilla, Sebastián Ortega), derecha simulación 2d de proyección (Henry Urley Portilla)**.** [cuadro comparativo]

Si se compara la proyección realizada en el *workshop* con la proyección realizada con *Python* se puede observar la clara diferencia puesto que la proyección realizada en el *workshop* hace referencia a proyecciones estadísticas simples lineales. Mientras que las proyecciones realizadas en *Python* son tendencias analizadas como nubes de puntos con análisis avanzados como es este caso de la regresión lineal simple.

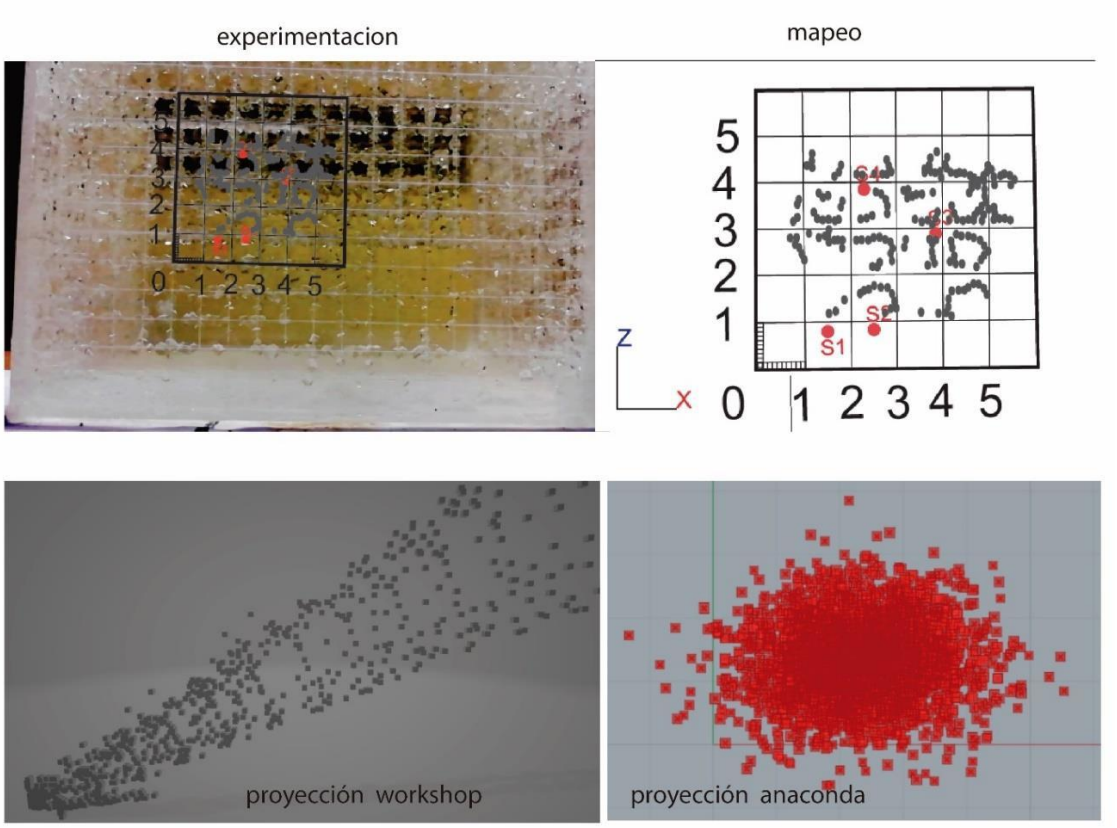


Figura 55 Agregación de partículas
Nota: Arriba mapeos 2d (elaborado, Jorge Hernández, Camilo Ramones), abajo izquierda simulación 2d de proyección *workshop* (elaborado, Jorge Hernández, Camilo Ramones), derecha simulación 2d de proyección (Henry Urley Portilla)**.** [cuadro comparativo]

En la imagen anterior, se puede observar la comparativa, principalmente en la proyección del *workshop* como esta repite un patrón y lo extiende linealmente mientras que cada agregación deja de estar aglomerada y se convierte en partes individuales. Sin embargo, la proyección realizada con anaconda y *Python* puede ser fácilmente comparada con la experimentación donde

se puede ver que estas se agregan y se aglomeran todas en un mismo espacio caso que es muy cercano a la realidad.

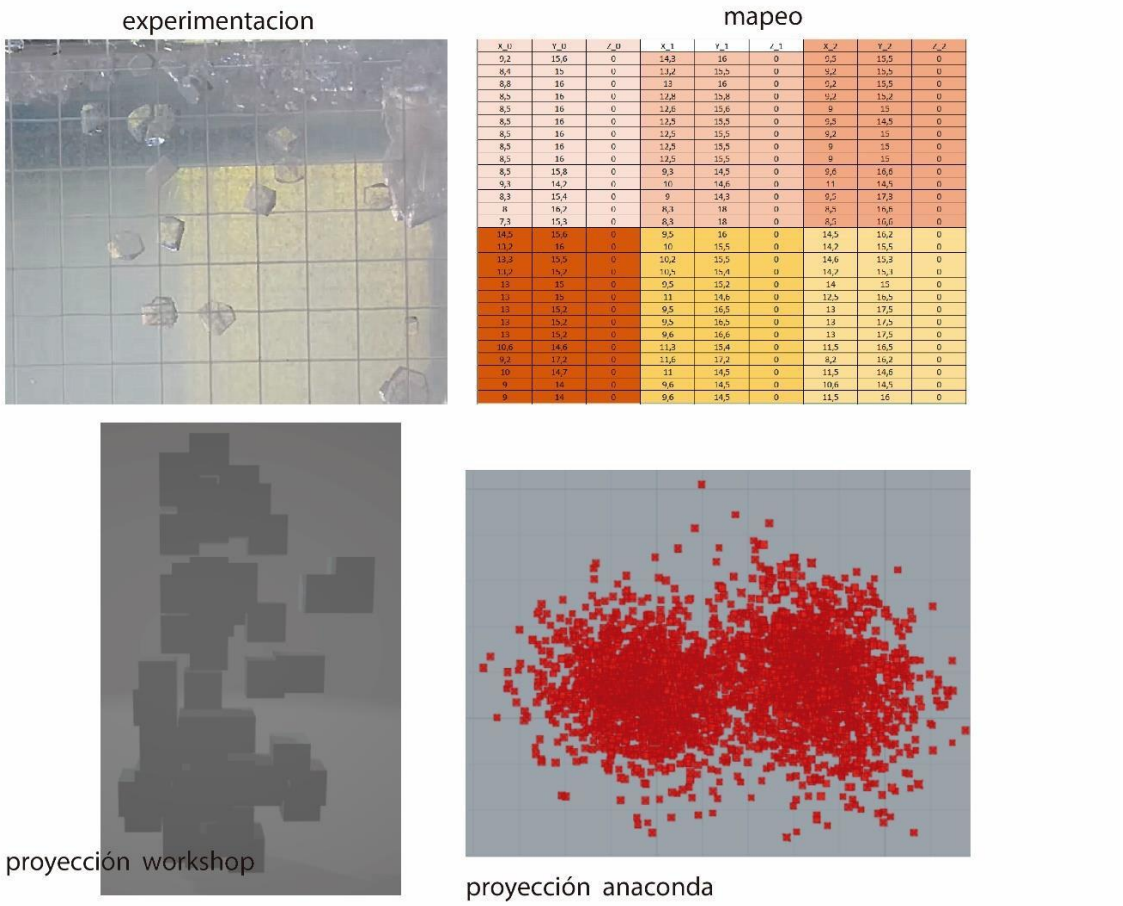


Figura 56 *Agregación de partículas*

Nota: Arriba mapeos 2d (elaborado, Eduardo Sandoval, Alejandra Tiria), abajo izquierda simulación 2d de proyección *workshop* (elaborado, Eduardo Sandoval, Alejandra Tiria), derecha simulación 2d de proyección (Henry Urley Portilla)**.** [cuadro comparativo]

La tendencia de crecimiento en las agregaciones bidimensionales consiste en la ocupación de superficies o el seguimiento de patrones en este caso perturbaciones por corte a laser, las partículas crecen de modo individual y las demás se van acoplando a la perturbación hasta ocupar el total de esta, esto se observa en tanto la simulación proyectada con *Python* como la experimentación. Partículas individuales que se aglomeran para la conformación de un todo.

Comparativas de Simulaciones Tridimensionales

Los datos tridimensionales son aquellos que fueron recolectados durante la experimentacion, partiendo desde el nacimiento o agregacion de particulas que se forman en el hilo susupendido en el centro del recipiente que contiene la solucion y a su vez el recipiente se encuentra marcado para la toma de coordenadas en los ejes (x, y, z). En este caso tridimensional el hilo suspendiso funciona como la perturbacion que provoca que las particulas se agreguen pero debido a que esta no posee una superficie plana hace que las particulas se agreguen a modo multidireccional dentro del espacio del recipiente y siguiendo el trayecto de la perturbacion.

Los datos recolectados de esta experimentacion son llevados al simulador donde se puede observar la agregacion de las particulas se manera cercana a la experimentacion, posteriormente a esto se realizo la proyeccion de los datos de manera analoga por cada participante del workshop donde el comun denominador fue el uso de la herramienta de tendencia excel, dando como resultado en la simulacion modelos lejanos de la realidad. Estos mismos datos fueron expuestos al analisis y proyeccion de python con el fin de obtener muestras para la visualizacion de posibilidades de agregacion de particulas y su orientacion.

La siguiente comparativa hacer refencia al grupo conformado por Cristina Lizcano y Jordin Suarez, cuyo mapeo y recoleccion de datos no solo se baso en la agregacion de particulas sino que tambien en el crecimiento de estas, lo que provoca que estos datos recolectados no pertenezcan solo la agregacion de particulas, pero los datos fueron ordenados de modo que sean interpretados como agregaciones.

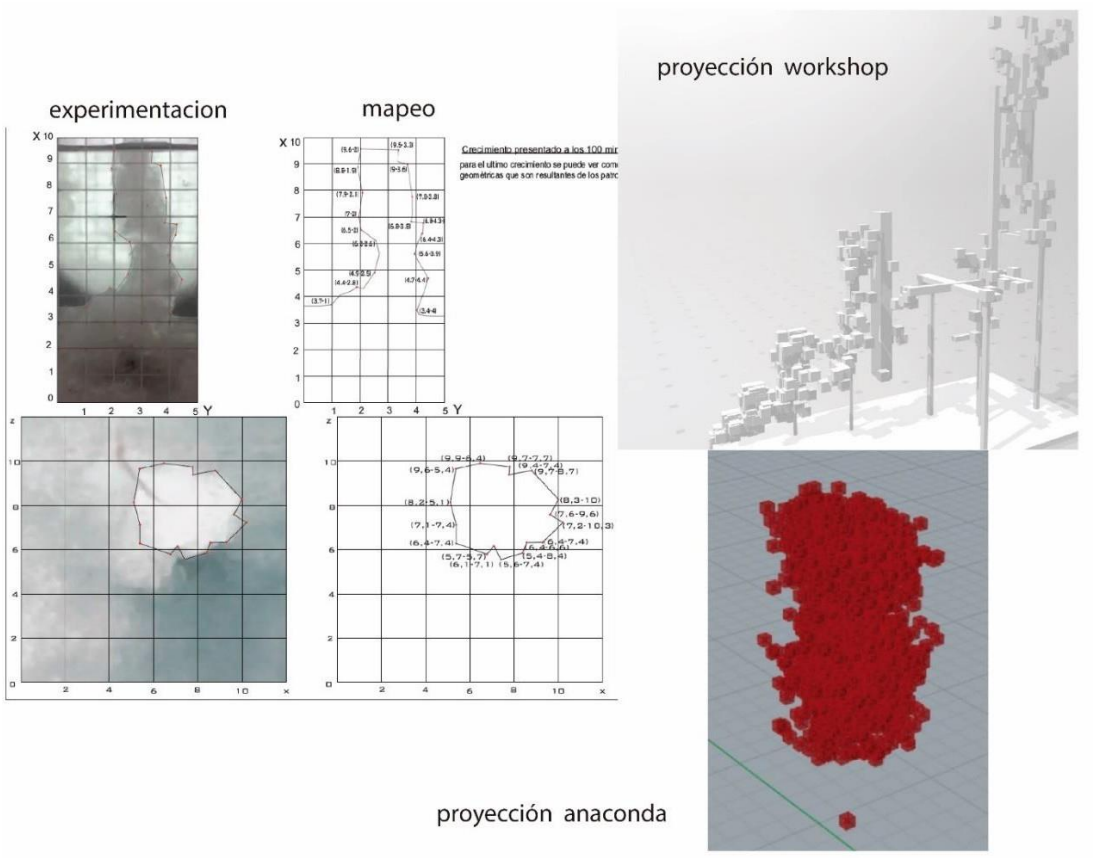


Figura 57 Agregación de partículas

Nota: Arriba mapeos 3d (elaborado, Cristina Lizcano, Jordin Suarez), abajo izquierda simulación 3d de proyección workshop (elaborado, Cristina Lizcano, Jordin Suarez), derecha simulación 3d de proyección (Henry Urley Portilla) [cuadro comparativo]

En la primera simulación se puede observar como las los volúmenes hacen referencia a las partículas agregadas, estos datos fueron llevados a el análisis con *Python* quien los interpreta como nube de puntos y son proyectados. Al visualizar los nuevos datos proyectados en el simulador, se pudo observar cómo fue las coordenadas de agregación y crecimiento fueron únicamente interpretadas como nube de puntos por *Python*, pero igualmente de esa manera se observa como esta tendencia es cercana a la realidad ya que el crecimiento de la estructura se da en grosor caso que pasa en la experimentación, la nueva visualización interpreta estos datos

como agregaciones y no como crecimiento de las partículas contrario a como fueron recolectados los datos.

En este segundo grupo de datos recolectados en el desarrollo del *workshop*, se realizó la agrupación de los datos en tres secciones principales donde desarrollaron las agregaciones en la perturbación tridimensional, los datos fueron tomados como los demás grupos solo por agregación de las partículas y no por crecimiento. Los datos fueron ordenados en las tablas *Excel* que posteriormente se visualizaron y proyectaron con el uso de *Excel* dando como resultado une estructura ramificados, pero lejos de lo que pasaría con la realidad de las agregaciones. Los datos obtenidos en la experimentación por Jhon Mantilla y Sebastián Ortega, fueron exportados al código *Python* para su análisis y proyección.

Los resultados obtenidos de esta proyección son simulados de los cuales se obtiene tres nubes de partículas que a simple vista parecen aglomerados, una vez revisados las agregaciones se encuentran separadas. Sin embargo, se pudo observar que si existe una similitud en la agregación de partículas comparado a la experimentación directa con la materia.

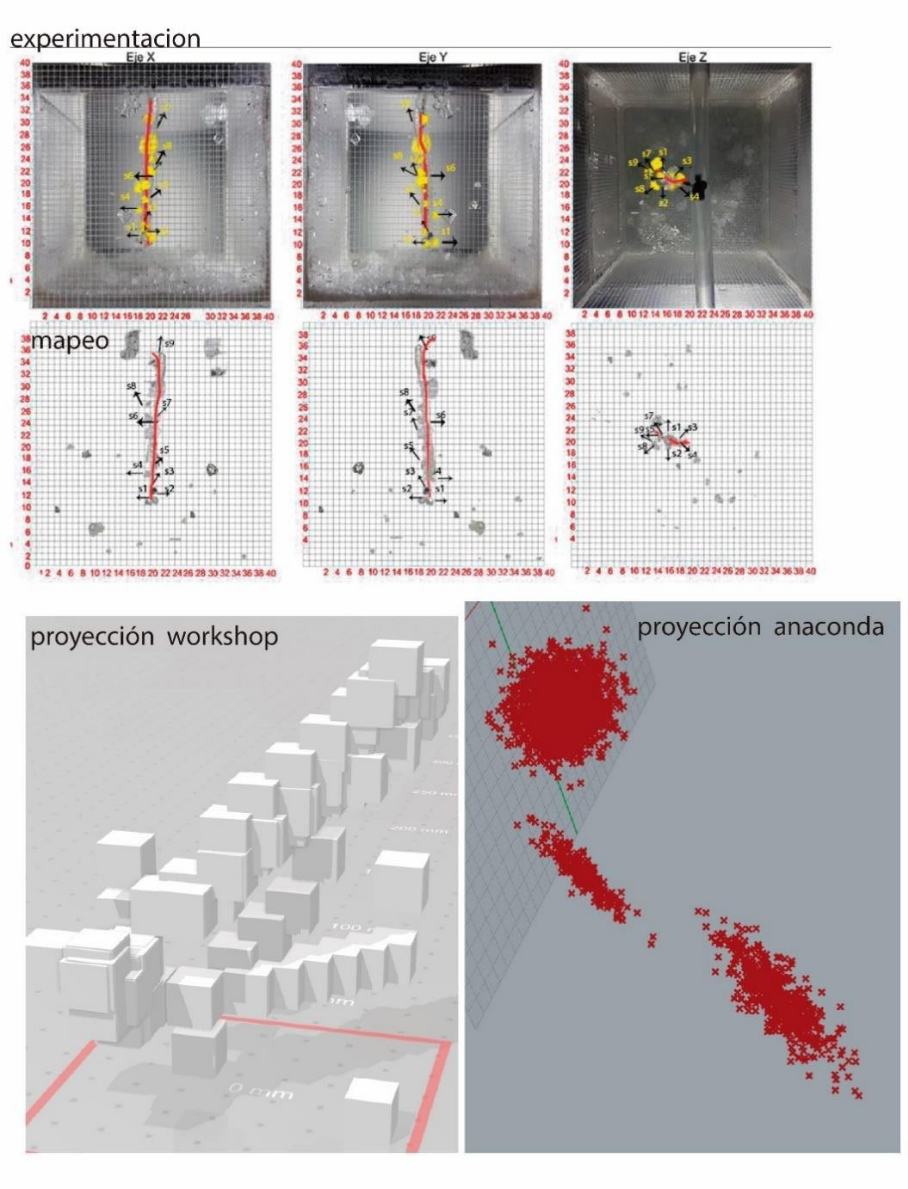


Figura 58 Agregación de partículas

Nota: Arriba mapeos 3d (elaborado, Jhon Mantilla, Sebastián Ortega), abajo izquierda simulación 3d de proyección workshop (elaborado, Jhon Mantilla, Sebastián Ortega), derecha simulación 3d de proyección (Henry Urley Portilla). [cuadro comparativo]

La experimentación y mapeo realizado por Jorge Hernández y Camilo Ramones, se realizó con una captación detallada de datos que posteriormente fueron muy bien ordenados dentro de la tabla Excel, estos datos fueron visualizados en la simulación para luego ser proyectados de

manera análoga obteniendo resultado una estructura de partículas ramificada en donde están fuertemente marcadas en orientación muy lineal lo que lo aleja de la realidad.

Los datos recolectados fueron importados a *Python + anaconda* con el fin de ser analizados y proyectados desde el uso de herramientas de inteligencia artificial, de lo cual se obtuvo coordenadas equivalentes a 3000 nuevas agregaciones. Estos nuevos datos fueron simulados para resultar así la agregación de partículas de manera muy cercana a la realidad, ya que las partículas generaron una aglomeración similar al resultado final de la experimentación, pero esta vez con una dirección más amplia, se obtiene una proyección y una orientación de agregaciones que representa a la realidad.

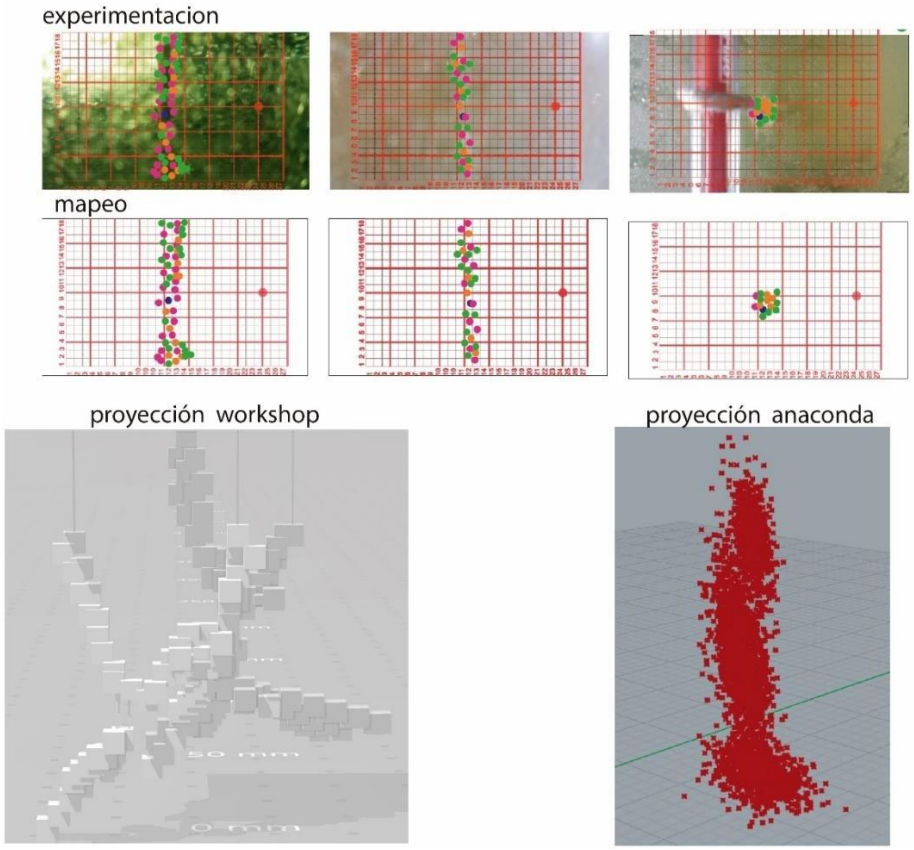


Figura 59 Agregación de partículas

Nota: Arriba mapeos 3d (elaborado, Jorge Hernandez, Camilo Ramones), abajo izquierda simulación 3d de proyección workshop (elaborado, Jorge Hernandez, Camilo Ramones), derecha simulación 3d de proyección (Henry Urley Portilla). [cuadro comparativo]

Esta última comparación entre la experimentación y su proyección, hasta llegar a la proyección realizada con *Python + anaconda* es el resultado de la poca captación de datos, ya que durante el mapeo se organizaron solo tres coordenadas, lo que tiene como consecuencia que en el análisis solo se halle un patrón de agregación provocando esto que al momento de proyectar y visualizar las partículas se congreguen en un solo espacio establecido dentro de los tres datos originales.

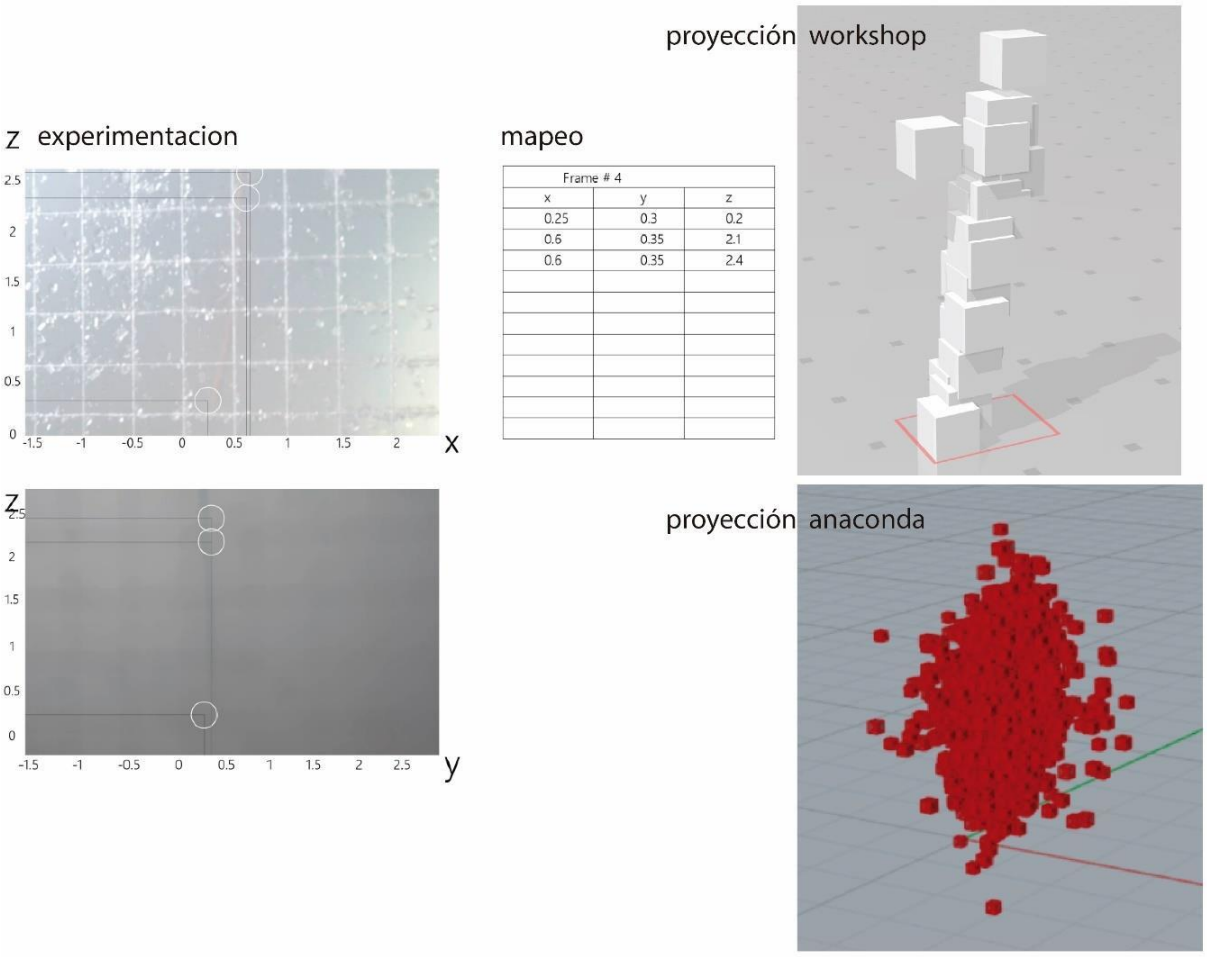


Figura 60 Agregación de partículas

Nota: Arriba mapeos 3d (elaborado, Eduardo Sandoval, Alejandra Tiria), abajo izquierda simulación 3d de proyección workshop (elaborado, Eduardo Sandoval, Alejandra Tiria), derecha simulación 3d de proyección (Henry Urley Portilla)**.** [cuadro comparativo]

A través de la simulación se pudo observar que, al igual como en las experimentaciones las estructuras cristalinas se forman con la aglomeración de partículas individuales que dan forma a un todo y ocupan el espacio perturbado en el recipiente. El fenómeno donde una partícula individual interactúa directamente con otras iguales para la conformación de un todo se le conoce como fenómeno de emergencia.

Cabe destacar, que es el caso del material cristalizado o la agregación de las partículas simuladas, ya que son elementos individuales que con la aplicación de energía provoca la aglomeración en espacios definidos y así forman estructuras. Las estructuras obtenidas de la simulación pueden ser analizadas desde sus partículas agregadas y el modo en el cómo construyen la estructura, para de esta manera entender el sistema dinámico de agregación de partículas que se encuentra presente en la simulación y experimentación.

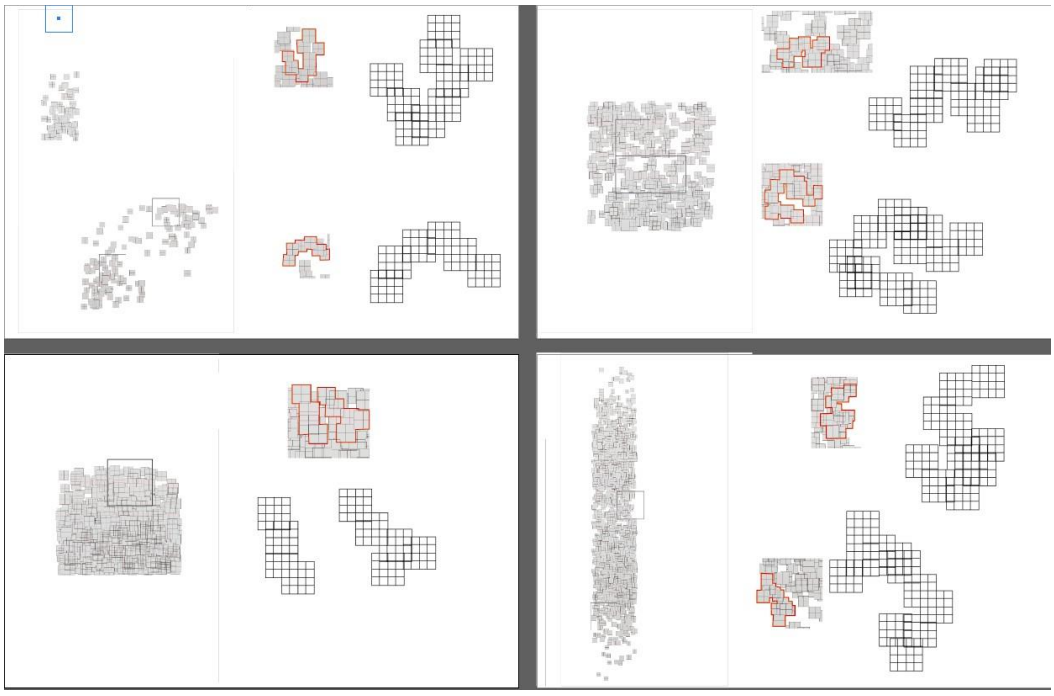


Figura 61 Muestras y análisis de las agregaciones simuladas [diagrama]

El anterior diagrama parte de la toma de muestras de las estructuras bidimensionales o tridimensionales obtenidas de las simulaciones, esto con el fin de comprender el fenómeno de

emergencia o como las partículas individualmente se van a agregando para la formación de estructuras relevantes. Por lo tanto, se creó un cuadrado que esta subdividido en 4 x 4 partes iguales, este hace referencia a los cubos o las partículas que conforman las estructuras.

Con el uso del cuadrado se buscó generar una réplica de las muestras tomadas de la estructura, esto con el fin de hallar o entender como estas piezas se aglomeran entre ellas. Esto dio como resultado que las partículas agregadas individualmente, ocupan 1/4 o 2/4 sobre la siguiente partícula que se agrega. Este porcentaje de ocupación se da en cualquier cara de la agregación, pero nunca se ocupan todas las caras de la partícula agregada incluso si esta se encuentra en la parte céntrica de la estructura.

El análisis dio como resultado el diseño de una pieza cubica cuyas caras se encentran dividida en dos partes de las cuales una de esta ocupa 1/4 de cada cara igual como sucede en la agregación de partículas simulada. Este ejercicio de análisis surgió de la búsqueda de la comprensión de la agregación de las partículas individuales para la conformación de las estructuras relevantes de la simulación.

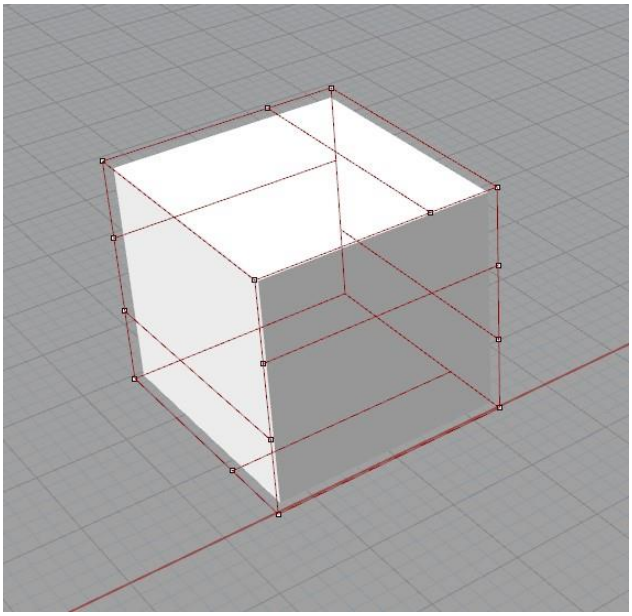


Figura 62 Cubo diseñado basado en el análisis de agregaciones simuladas [modelo]

El código está basado en la definición en grasshopper de agregaciones de *Gediminas Kirdeiski* que tiene como objetivo controlar las caras por las que generara la agregación de partes para la construcción de estructura. La definición fue adaptada en este caso con el fin de poder visualizar lo comprendido en el análisis de las agregaciones simuladas y como estas conforman y construyen estructuras.

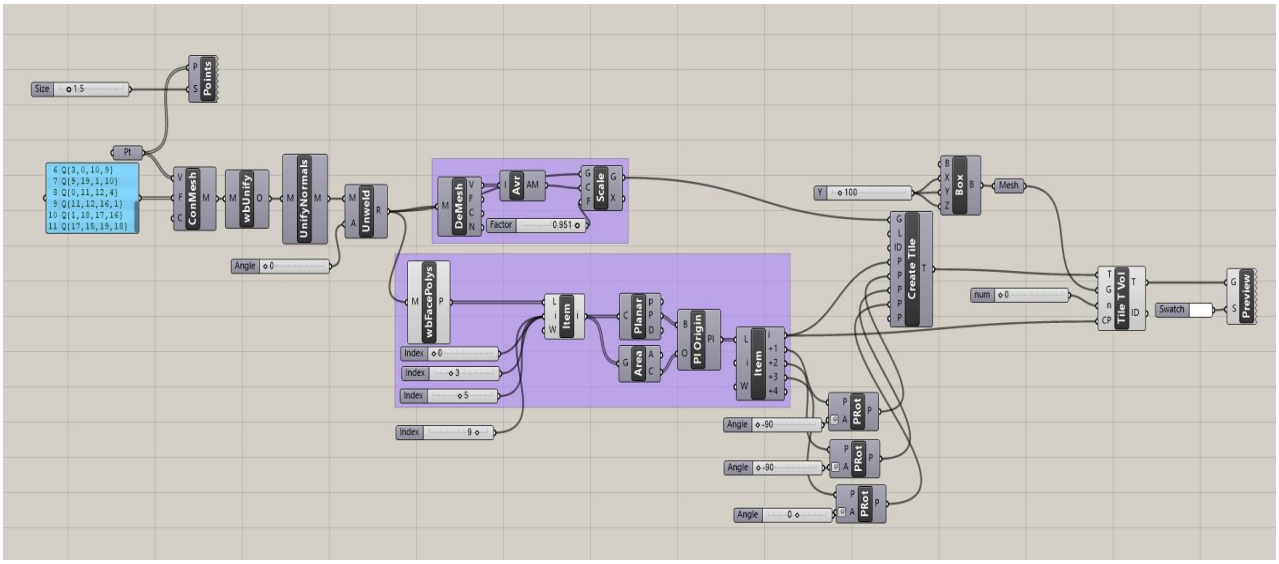


Figura 63 Simulador de agregaciones basado en *Gediminas Kirdeiski* [definición]

Tras el funcionamiento de la definición de *grasshopper* se obtiene como resultado una estructura similar en resultados a los obtenidos de la simulación de partículas proyectadas que pueden ser comparados, e incluso ocurren sucesos similares al fenómeno de emergencia. Tras la investigación se encontró que derivado de este fenómeno de emergencia se encuentra dos términos que definen los fenómenos vistos tanto en la experimentación directa con la materia o la simulación proyectada estos son el auto ensamblado y la auto organización.

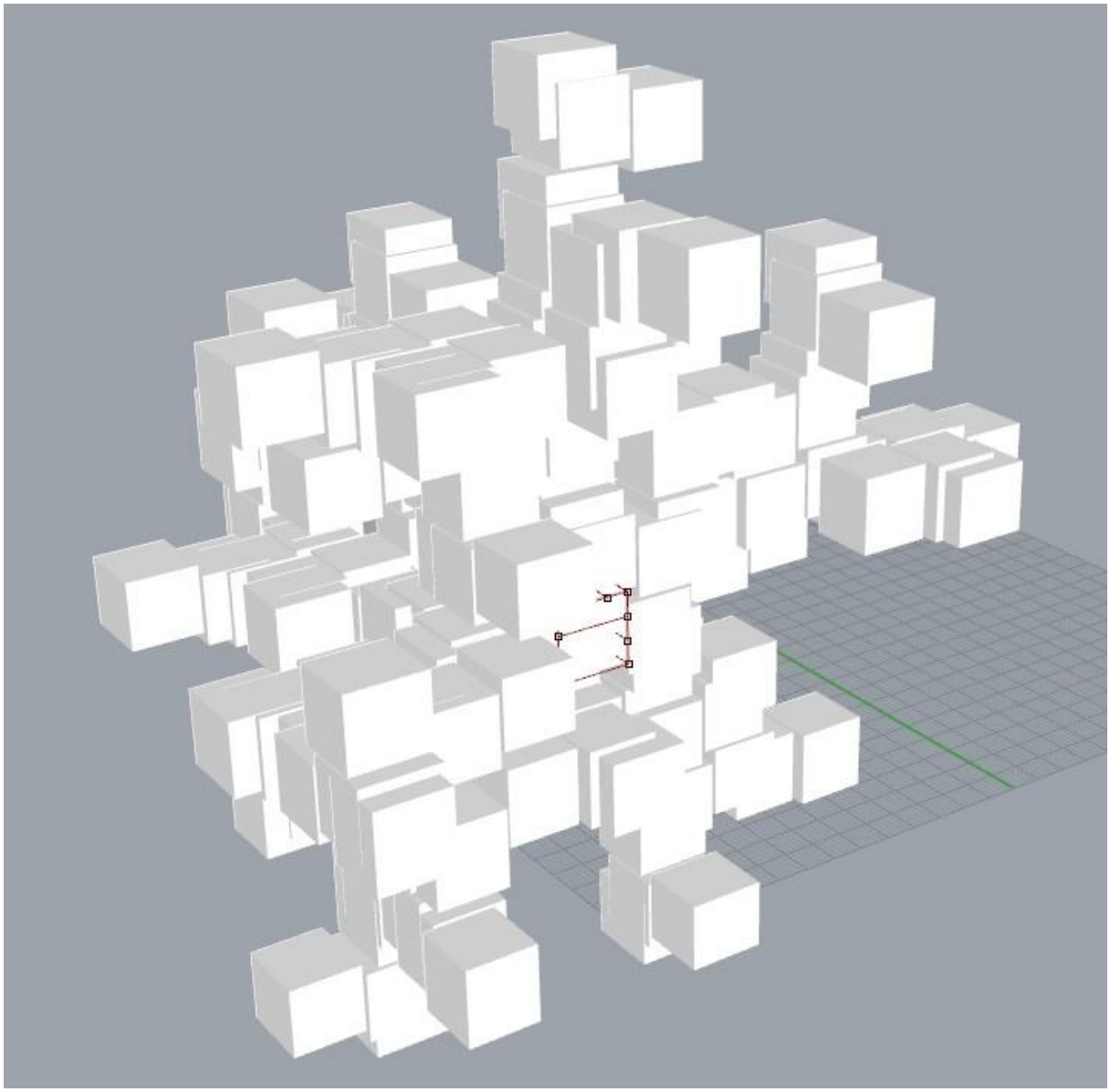


Figura 64 Resultado de reproducción de la simulación. [modelo]

Estos dos términos tienen un uso de origen en el mundo natural y estos fenómenos que ocurren a un nivel que va de micro a macro, actualmente son usados por dos referentes relevantes como lo es Skylar Tibbits y Arthur Olson quienes desarrollan temas referentes con la modelización y fabricación de partes que se auto construyen o auto ensamblan. Estos dos conceptos anteriormente mencionados funcionaran como basas para la modelización de

estructuras relevantes cuya formación sea similar o replique la agregación de partículas observados en la experimentación o visualizados en la simulación proyectada.

5.5.3 Conclusión. La simulación los datos proyectados permite tener una visualización completa sobre la posibilidad como se agregarán las partículas si se sometiesen a una mayor concentración de materia en la solución y esta fuese expuesto a un tiempo de espera más amplio. Sin embargo, se debe tener en cuenta que la simulación de los datos proyectados es tan solo una posibilidad en la futura realidad por lo tanto no es de total fiabilidad. Por esto, se realizó la comparación de la experimentación y los modelos obtenidos de las simulaciones.

De la comparativa se pueden realizar sugerencias para la próxima toma de datos en las experimentaciones, principalmente una mayor captación de coordenadas porque de esta depende que tan viable sea la agregación de partículas proyectada y simulada. Adicionalmente, la comparativa permite estudiar el proceso de agregación de partículas y como los volúmenes individuales se van adicionando entre sí para la formación de las estructuras cristalinas. De esta forma, se pueden empezar a denominar los múltiples fenómenos presentes en la cristalización como lo es fenómeno de emergencia, la aplicación de energía, la auto construcción y sus derivados como lo es el auto ensamblado y la auto organización.

Conjuntamente, se realizó la indagación de referentes que exploran los términos anteriormente mencionados como lo es Arthur Olson y Skylar Tibtis, esto conllevó a que, tanto los términos adquiridos en la comparación y en conjunto a los referentes, se realice una próxima exploración sobre la aplicación de los fenómenos de la cristalización para la modelización de nuevas estructuras relevantes.

5.6 Modelización de Estructuras Auto Organizables

5.6.1 Introducción. Una vez comparadas las proyecciones y analizados con cada una de las experimentaciones para comprobar su viabilidad se obtuvieron conceptos o bases para la modelización de estructuras relevantes. Cabe recordar las bases conceptuales para la modelización de estructuras relevantes que surgieron del análisis de los modelos simulados de datos proyectados con *Python + anaconda* de agregaciones de partículas cristalina.

En conjunto con el análisis de modelos y agregaciones se hallaron los referentes Arthur Olson y Skylar Tibbits quienes tratan temas como el auto ensamblado y la auto organización, estos términos son derivados y nacientes del ya nombrado fenómeno de emergencia. Estos referentes tratan temas sobre la materialización y modelización de estructuras con términos que comúnmente son observados en la naturaleza, pero que a través de su estudio buscan generar nuevos métodos de autoconstrucción basándose en los fenómenos de auto ensamblaje y auto organización.

La auto organización y el auto ensamblado hacen referencia a la agregación de partículas para la materialización de nuevas estructuras, inicialmente estos conceptos son similares, pero en sus resultados de agregación son diferentes. El auto ensamblado ocurre cuando las partes o partículas de un todo son sometidas a una dinámica, pero cuyo resultado siempre será una forma o estructura final establecida. Mientras que la auto organización, es cuando unas partículas son sometidas a una dinámica y las piezas tienen la capacidad de auto acoplarse, construirse, destruirse y reconstruirse cuyo resultado final son estructuras relevantes nuevas.

5.6.2 Desarrollo. Los conceptos nombrados en la introducción de esta etapa se diseña un ejercicio o experimento cuya función es comprender la teoría de los dos términos, así que, el experimento parte de la idea de la aglomeración de particular para la obtención de estructuras, por lo tanto, primero se realiza la exploración de algún tipo de objeto donde las partes se puedan añadir y generar resultados, esto dio con hallazgo del clásico juego de "*jazz* o *matatena*".

Las piezas de este juego contienen características en su forma tridimensional abierta, ya que estas piezas pueden interactuar directamente con demás partes para de esta manera acoplarse y agregarse entre ellas en busca de obtener resultados. El principal motivo por el cual la pieza de *jazz* es capaz de agregarse se debe a su forma de cruceta abierta, ya permite la agregación de uno o más *jazz* dentro de la pieza inicial, otra característica de esta es llevar cuatro esferas en cuatro de sus seis puntas, esto permiten que exista un nivel de acoplamiento entre las demás piezas que se agregan a la inicial.

Figura 65 Piezas o partículas del juego jazz [fotografía]

Partiendo de las características anteriormente nombradas se diseñó un primer ejercicio para la formación de estructuras con partículas agregadas. El ejercicio o experimento consiste en verter 180 partículas en un recipiente y observar la posibilidad de formación de estructuras. Este primer ejercicio no se obtuvo ningún tipo de resultados de estructuras puesto que las piezas al ser abiertas, rápidamente se desmoronan su estructura en formación, pero si se pudo ver como en los puntos más cerrados de este recipiente las partículas de jazz se aglomeraban y se adaptaban a ese espacio.

Por ende, se creó otro ejercicio que consiste verter todas las 180 partículas en un recipiente esta vez uno más reducido en donde las piezas de jazz se encuentren más próximas entre ella, para posteriormente en un rápido movimiento girar el recipiente sobre una superficie plana, para luego retirarlo y así observar cómo las piezas toman la forma del recipiente o entorno donde se encuentran. Los resultados de este ejercicio son que al retiro del recipiente las piezas se desplomaron y la forma no se conservó. Al realizar por una segunda vez este ejercicio el recipiente no fue retirado una vez girado, sino que se le aplico energía.

La energía aplicada a las partículas al interior del recipiente, consiste en realizar movimientos oscilantes o circunferenciales en tramos estrecho para generar una sensación de vibración, una vez hecho esto se aplica vibración dando pequeños golpes a las paredes laterales del recipiente. Una vez realizados los movimientos anteriores, se puede observar a través del recipiente como el volumen de las piezas se ha reducido, se realiza el retiro del recipiente. Se obtiene como resultado la formación de una estructura cuya forma pertenece a la del recipiente.

Figura 66 Piezas de jazz dentro del recipiente segundo ejercicio. [fotografía]

Esto se debe a que las partículas del *jazz* se adaptaron a la forma del recipiente y al realizar la aplicación de energía sobre las piezas esta se compactaron y se agregaron de tal forma que no dejaron espacios vacíos o puntos débiles provocando de esta manera que tomaran la forma del recipiente como forma establecida.

Figura 67 Adaptación del *jazz* a su entorno. [fotografía]

Lo sucedido en este ejercicio puede ser comparado con el auto ensamblado, ya que las piezas de *jazz* son partículas individuales que se encuentran en un entorno establecido y que al aplicar la energía, las partículas se construyen para obtener una forma final establecida como lo es en este caso el recipiente. Tratando el tema de cristalización el proceso de auto ensamblado, ocurre cuando molecularmente las piezas se construyen con una forma establecida en este caso los cristales de alumbre sus partículas cristalinas tiene formas semi triangulares.

Figura 68 Partículas de cristal de alumbre [fotografía]

Basado en el auto ensamblado y la adaptación a formas establecidas se realizó un tercer ejercicio que consiste en la adaptación de forma con la aplicación de vacíos.

El ejercicio consiste en la aplicación de un vacío (globo) dentro del recipiente en el que se verterán las piezas de jazz con el objetivo de que las partículas se adapten a las formas establecida por el entorno. Una vez realizado este tercer ejercicio se obtuvo la adaptación de las piezas de *jazz* al vacío, una vez las piezas son compactas en el entorno establecido se procede hacer el retiro del globo, del cual se puede observar cómo las partículas de adaptaron al espacio vacío dejando la forma sin desmoronarse.

Figura 69 Arriba, globo o vacío. Abajo, vacío resultante [fotografía]

Esta nueva estructura resultante no puede ser extraída del entorno del recipiente debido a que esta se desplomaría. El principal motivo por el cual las estructuras materializadas por las piezas de jazz no son resistentes se da por la misma forma tridimensional de una cruceta abierta, lo que provoca que la pieza aglomerada pueda desacoplarse fácilmente al recibir incluso un leve movimiento. Por lo tanto, partiendo del principio de agregación de la pieza de jazz se pretende

generar o rediseñar a esta con el fin de dar con una pieza con la capacidad de agregarse o acoplarse y menor probabilidad de desacoplarse.

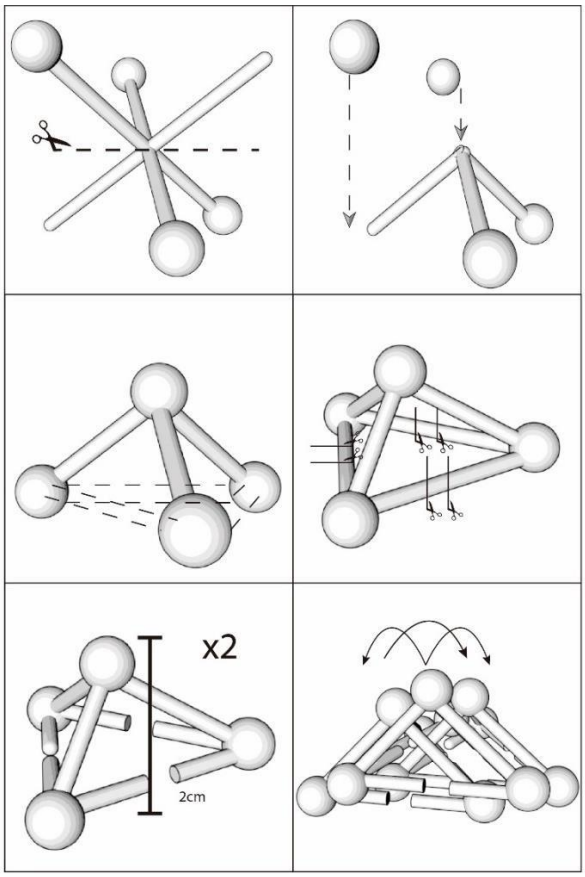


Figura 70 Diseño de pieza de agregación [diagrama]

Se diseñó una pieza de agregación tetraédrica, en cuyos vértices se encuentra una esfera en función de servir como encastre o seguro de las piezas que se aglomeren a esta, tres de las aristas que componen esta figura se encuentran fraccionadas con el objetivo de generar una abertura equivalente al diámetro de las aristas del tetraedro, la función de esto es poder agregarse a otras piezas y no tender a separarse. Una vez diseñada la pieza se procede a su fabricación que se realiza a través de la impresión 3d con filamento PLA.

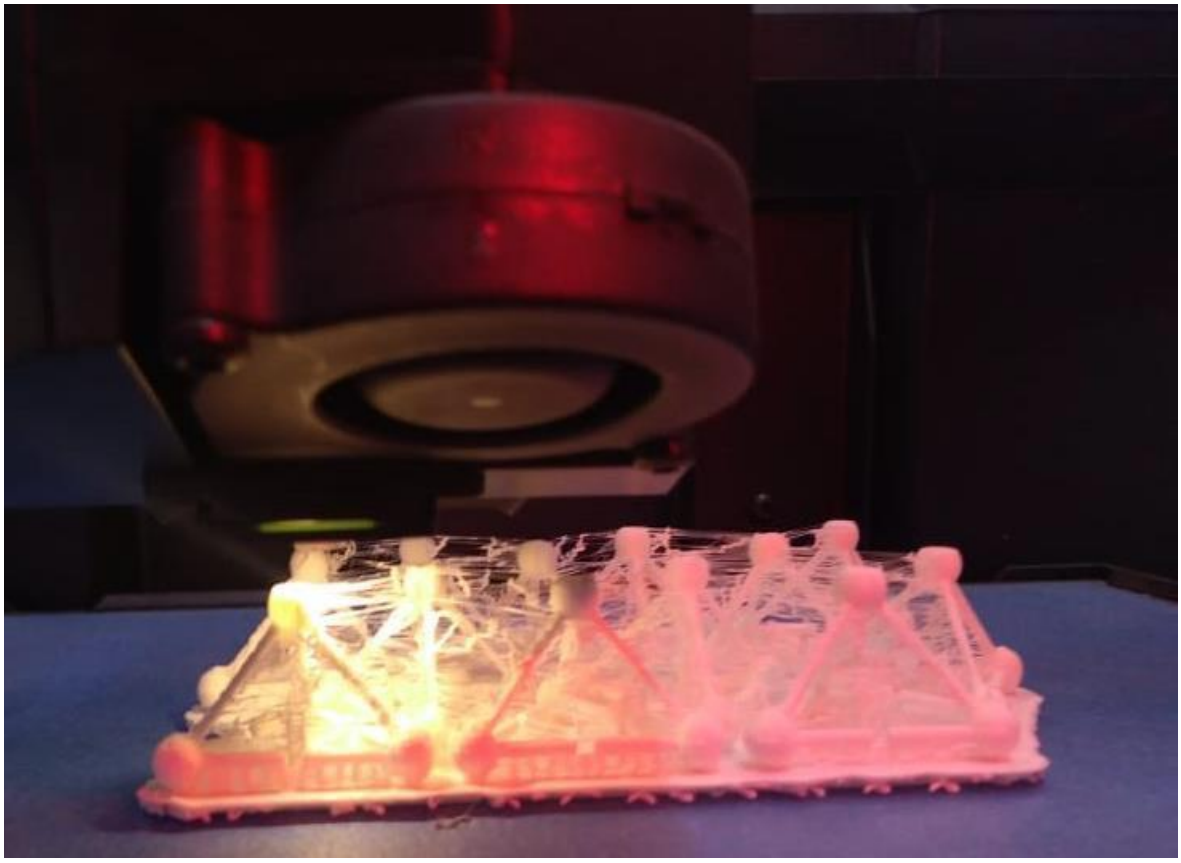


Figura 71 Impresión 3d de piezas [fotografía]

Se fabricaron un total de 12 piezas, pero al momento de ser despegadas de la base donde son fabricadas 4 piezas se quebraron quedando así solo 8 funcionales. Con las piezas disponibles se realiza un nuevo ejercicio que consiste en colocar dentro de un recipiente con tapa las nuevas piezas, una vez dentro se debe sellar el recipiente y someterlo a movimientos oscilantes y circunferenciales, constantes durante un tiempo para obtener resultados de agregación de

partículas.

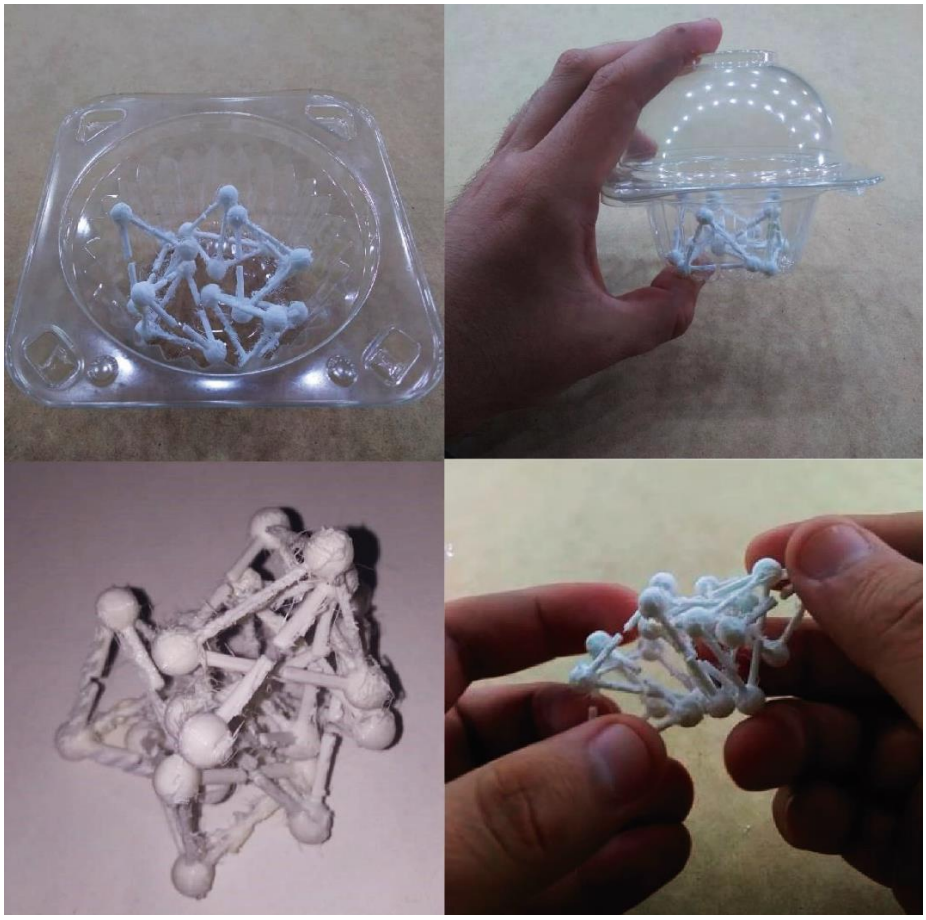


Figura 72 Ejercicio de agregación con nuevas piezas. [fotografía]

Las piezas fueron sometidas a la aplicación de energías o dinámicas durante lapsos de seis minutos en dos ocasiones, de los cuales en cada ocasión resulto una estructura nueva, la primera estructura agregó todas las piezas puestas en el recipiente dando resultado una única estructura de la cual se podía destacar como se agrupaban todas piezas mutuamente creando una forma más compacta, mientras, que la segunda estructura se puede observar como las piezas se enlazaron en forma de cadena. Durante la ejecución del ejercicio se puede observar como las piezas en el recipiente tienen la capacidad de agregarse, destruirse y reconstruirse en múltiples estructuras y al finalizar la aplicación de la energía, observar resultados de dos estructuras distintas en dos

ocasiones durante el desarrollo de este ejercicio, se puede decir que estas son estructuras auto organizables.

Sin embargo, el diseño y la fabricación de las piezas presento errores como la elección del material, ya que la aplicación de energía al momento de realizar los ejercicios de auto organización las piezas tetraédricas se daña. En base al diseño y función de la pieza auto organizables tetraédrica se estableció la implementación de una nueva pieza con funciones similares, pero que esta vez la pieza tenga las características de atrapar o encastrar.

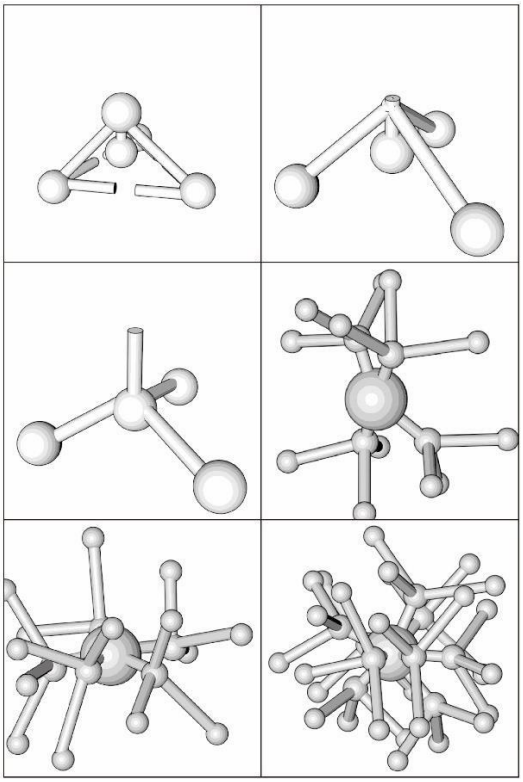


Figura 73 Diseño de pieza de agregación partiendo de tetraedro [diagrama]

La nueva pieza diseñada consiste en un elemento esférico central del cual se derivan unas ramificaciones tetraédricas que giran en su propio eje y se distribuyen por la esfera central, para que sus partes complementen los espacios vacíos de esta esfera. La función de esta nueva pieza es generar agregación de piezas modulares, pero en modalidad de “atrapar”, esto quiere decir que

las piezas al someterse a la aplicación de energía tengan la capacidad de encastrarse entre las ramificaciones para de esta manera generar estructuras auto organizables. La pieza fue fabricada en impresión 3d con filamento PLA, debido a su diseño más compacto la pieza es más resistente y no sufrió daños, se realizó un ejercicio que consiste en colocar las piezas dentro de un recipiente y aplicar la energía.

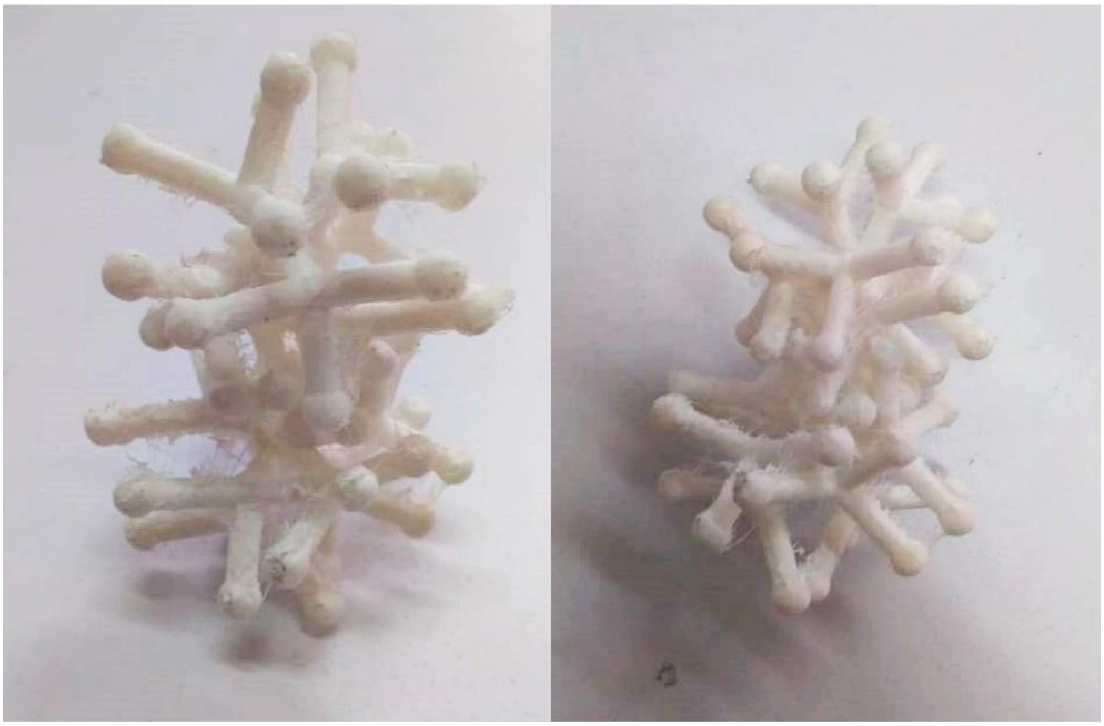


Figura 74 Ejercicio piezas esféricas. [fotografía]

Se comprobó el funcionamiento de esta pieza y su posibilidad de agregación para la modelización de estructuras auto organizables, esta pieza tiene la capacidad de atrapar a las demás piezas debido a la distribución de sus ramificaciones las cuales se encajan de esta manera se pueden agregar más piezas para la formación de estructuras auto organizables. Esta pieza tiene la característica auto organizable ya que al ser sometida a un entorno dinámico tienen la capacidad de construcción, acoplándosele y desacoplándose en múltiples ocasiones hasta que el entorno dinámico lo determine.

Una vez en este punto del desarrollo de este proyecto de pasantías se realizó la participación en el “primer congreso internacional de ALEPH Asociación Latinoamericana de Estudios de la Forma Universidad Nacional Pedro Henríquez Ureña” en Santo Domingo, capital de República Dominicana. La participación se realizó en modalidad poster, que fue realizado en conjunto con Ismael García (ponente sobre la cristalización), Juan Manuel Villa (tutor de pasantía), Ramón Galvis Centurión (cotutor de pasantía) y Henry Portilla Delgado (pasante), el poster llevó el título “Diseño y materialización de piezas para la auto construcción basado en la agregación de partículas”

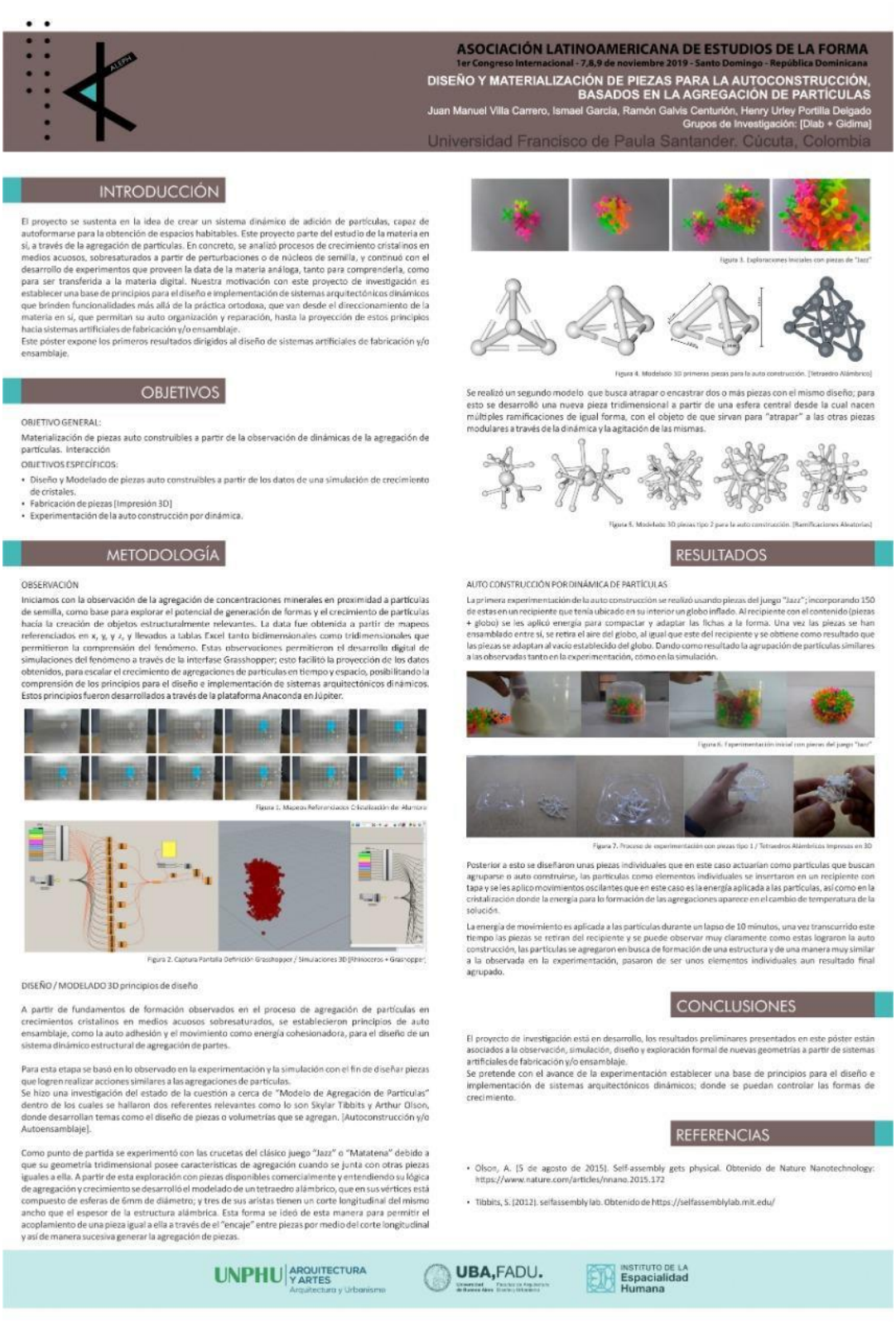


Figura 75 Poster para el primer congreso internacional de ALEPH Asociación Latinoamericana de Estudios de la Forma Universidad Nacional Pedro Henríquez Ureña
Fuente: [poster] elaboración Ramón Galvis, Juan Manual Villa, Henry Portilla.

Retomando la pieza tetraédrica se desea realizar un ejercicio con esta, pero primero se debe realizar un rediseño y materialización de esta ya que la primera pieza usada en los primeros ejercicios contenía errores tanto en diseño como de material.

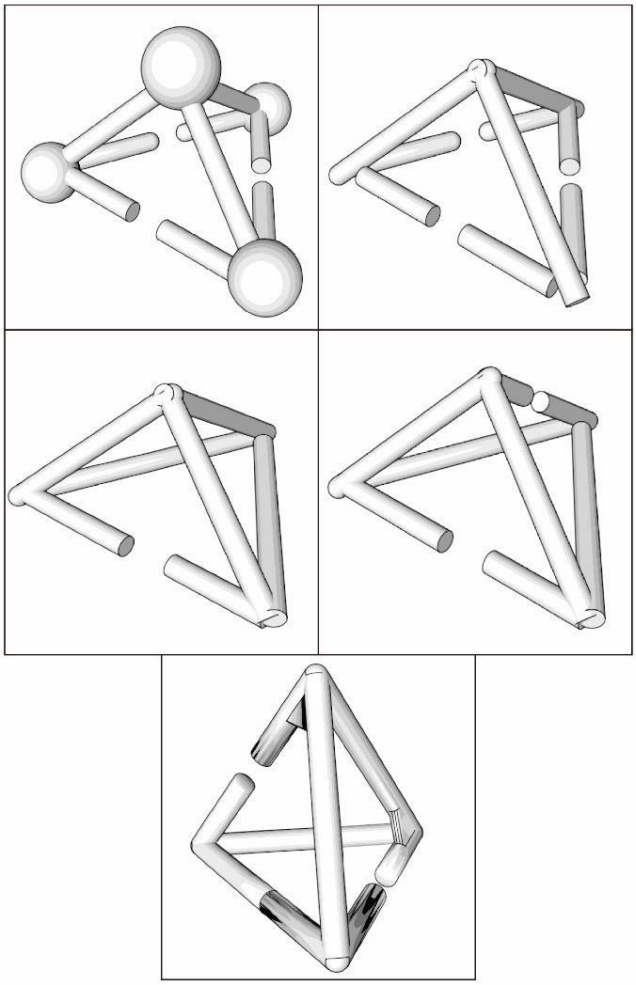


Figura 76 Rediseño de pieza tetraédrica. [diagrama]

La segunda versión de la pieza tetraédrica auto organizable, consiste en el retiro de las esferas ubicadas en los vértices de la pieza inicial ya que esta no posee la misma función de atrape que las crucetas de jazz ya que esta característica se encuentra presente en su misma forma. Las aberturas en la pieza auto organizable se reducen a dos, ubicadas en aristas paralelas y además de

añade un refuerzo en la cara más próxima a la abertura, este consiste en una pequeña pieza triangular ubicada en una esquina de la cara de la pieza tetraédrica.

Se puede señalar, que el diseño de las piezas individualmente hablando se realizó de modo modular, en el que cada grupo de nuevas piezas son idénticas entre ellas siguiendo el artículo de Arthur Olson llamado "*Self-Assembly Gets Physical*" el cual trata en una de sus párrafos, la probabilidad de combinación y de construcción es mayor cuando las piezas son similares entre ellas, mientras que si piezas son diferentes entre ellas es casi imposibles lograr un resultado final o la probabilidad disminuye en gran porción.

Por esto se diseñó una pieza por cada grupo de módulos, pero teniendo en cuenta la geometría tridimensional de la pieza al momento de interactuar con las demás en la búsqueda de la construcción de estructuras auto organizables. De esta manera se obtienen piezas cuya mayor probabilidad en la construcción de estructuras auto organizables, basado en principios de la autoconstrucción y principios geométricos que partieron del juego clásico jazz, donde se puede observar cómo sus piezas similares entre ellas.

Primero se realizó un ejercicio para comprobar la funcionalidad de estas nuevas piezas posteriormente a su fabricación en este caso se usó la impresión 3d con filamento PLA duroplast material que es más resistente que el PLA tradicional. Entonces, las piezas fueron puestas en un entorno dinámico para comprobar la auto organización de las nuevas piezas.

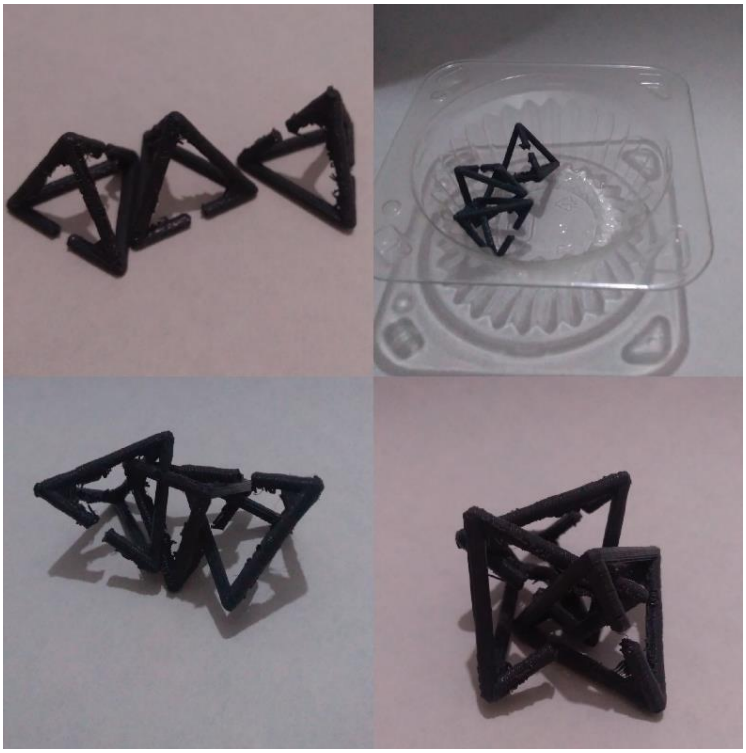


Figura 77 Ejercicio de auto organización de pieza tetraédrica. [fotografía]

De este ejercicio de prueba se obtuvo la auto organización con un patrón de ensamblado similar a la primera pieza tetraédrica auto organizable de tres aberturas. La primera el acoplamiento en cadena y el segundo cuando tres piezas se auto organizan acoplándose mutuamente entre ellas para dar un resultado más estructural en su forma. Partiendo de los ejercicios con la pieza tetraédrica y las esféricas ramificadas se realizará un ejercicio donde se pueda observar la agregación de partículas para la formación de estructuras auto organizables, basado en las estructuras formadas en las simulaciones de datos proyectados de agregaciones cristalinas.

Este ejercicio consta del uso de las piezas tetraédricas dispuestas de dos maneras distintas en un mismo entorno dinámico. La primera parte consiste en el uso de la pieza tetraédrica auto organizable y la segunda parte es un elemento vertical, el cual tiene piezas tetraédricas que están

dispuestas a lo largo de este y funciona como elemento donde se agregaran las piezas auto organizables para dar forma a una estructura. El objetivo de este ejercicio es obtener estructuras auto organizables cuya agregación se pueda comparar con los modelos obtenidos durante la simulación tanto con los datos proyectados como los de la experimentación cristalina, con el fin de entender la orientación, la agregación y los fenómenos que ocurren en la agregación de partículas cristalinas.

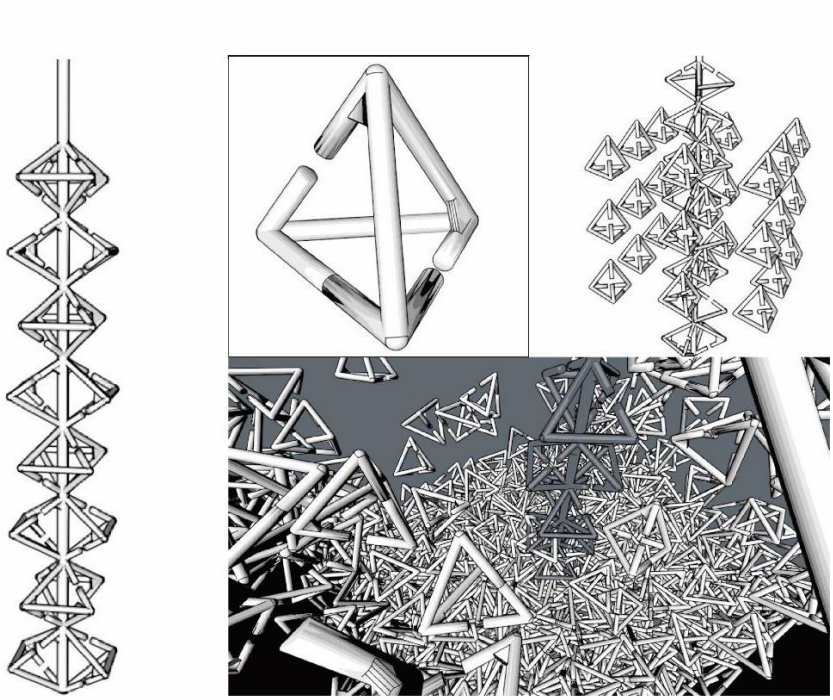


Figura 78 Ejercicio de estructuras auto organizables. [diagrama]

5.6.3 Conclusión. La simulación como punto de partida para la modelización de estructuras auto organizables es primordialmente quien define bases de los fenómenos esto comparado paralelamente con la experimentación esto con el objetivo de materializarlo. Definir cada uno de estos fenómenos profundiza su concepto y su función al momento de generar la agregación de partículas, por ende, entender esto ayuda a definir los modelos en la modelización de las estructuras.

Para dar un punto de partida a la creación de partes con características auto organizables se acude a la exploración o búsqueda de un elemento cotidiano cuyas características de adición sean similares. En este caso este elemento fue el juego de jazz cuya forma tridimensional se acopa a las demás piezas similares a la inicial. Al ser sometidas las piezas de *jazz* a pruebas se puede observar como esta se añade y se adapta al vacío del entorno dinámico (auto ensamblado) formando así estructuras cuya sostenibilidad es muy débil.

De esto se concluye que las nuevas piezas deben poseer la característica de añadirse a otra con tendencia a no desarmarse cuando se añade a otra partícula caso similar que ocurre cuando se experimenta directamente con la materia. Los nuevos modelos de piezas, al ser sometidos a pruebas dan resultado de la agregación de partes iguales, modelizando entre ellas estructuras propias del entorno dinámico y sus características físicas (auto organización).

Las piezas dentro de un entorno dinámico crean estructuras auto organizables fenómenos similares a los observados durante la experimentación directa con el material que agrega partículas cristalinas. Si se compara la creación de las estructuras auto organizables y la cristalización se puede contrastar desde la aplicación de energía en las piezas que funciona como el método para que estas piezas se agreguen entre ellas, caso similar sucede en la cristalización donde la energía que provoca la agregación de las partículas se produce en la aplicación y cambio de temperatura en la solución de materia cristalina.

El fenómeno de emergencia ocurre en las estructuras auto organizables cuando se agregan las piezas individualmente en un entorno y debido a las condiciones de la energía del entorno las partes estas se agregan de tal forma que emerge o se auto organiza una nueva estructura. Caso similar sucede en la cristalización cuando se crean partículas y debido al cambio de temperatura

se crea energía que provoca la unión y la creación de estructuras según el entorno que está ocupando, ya sea porque se agregan a las paredes o al filamento en el recipiente.

Se puede tratar igualmente el tema del auto ensamblado que está presente en las piezas de jazz que se agregan, se compactan y se adaptan a la forma del entorno establecido dando como resultado su misma forma de este, caso similar ocurre en la cristalización, pero esto se da en la construcción molecular del cristal que se forma y crea partículas con formas triangulares similares visibles.

6. Dificultades y Conclusión

6.1 Dificultades Presentadas Durante el Desarrollo de las Pasantías

Funcionalidad el código *Python* entregado durante el *workshop*, para usarlo como herramienta de proyección.

Falta de partes en el código *Python* entregado en el *workshop*.

La simulación como proceso de modelización de estructuras auto organizables.

Exploración de elemento con características de agregación para comparar con partículas cristalinas.

6.2 Estrategias de Solución a las Dificultades

Para llegar a entender el código *Python* se requirió de la investigación principalmente sobre las bibliotecas de anaconda usadas en el código, que son, para qué y cómo funcionan, para posteriormente determinar las partes que se usan específicamente de cada biblioteca en el código para que este funcione y tenga la capacidad de proyectar correctamente los datos importados. Una vez revisado el código es visible la falta de partes del código, estas fueron retiradas, esto provoca que el código no funcione arrojando error.

Se les dio solución a estas partes faltantes, investigando los tramos del código que arrojan error y comparándolos con códigos similares en internet o comparándolos los dos códigos entregados en el workshop, de esta manera se complementó el código. Los principales faltantes

en el código son la importación de tablas Excel, el análisis y graficación de datos tridimensionales, la exportación de los datos proyectados.

La simulación como proceso para la modelización de estructuras llevo a que se realizara el análisis sobre los modelos obtenidos de la simulación proyectada y de esta manera entender y determinar cómo las piezas se puede agregar entre ella para la formación de estructuras, e incluso obtener conceptos que refuerzan la comprensión del proceso de agregación de partículas cristalinas.

El juego clásico *jazz* permitió entender desde un elemento físico la construcción de estructuras agregadas, este sencillo juego permitió reforzar los conceptos de agregación en la simulación y de esta manera diseñar piezas que permiten obtener estructuras auto organizables.

6.3 Conclusiones

La simulación como proceso de modelización de estructuras auto organizables, sirve para entender y visualizar la agregación de partículas individuales que conforman una unidad o un todo en su entorno inmediato formando nuevas estructuras relevantes ya sean por datos que representan la realidad o una posibilidad de crecimiento en el futuro.

A través de las visualizaciones proporcionadas por la simulación, se pueden definir principios de formación de estructuras ya que en esta se puede observar desde una forma individual simple que construye y da forma, como es el cubo en este caso y partiendo de este se analizan las posibilidades de agregación y de esta manera definir reglas para el diseño de piezas que replique la modelización de estructuras auto organizables.

La modelización de estructuras organizables partiendo desde la simulación, es posible siempre y cuando se tenga en cuenta de donde surgen los datos que se están usando, como es en este caso la agregación de la materia cristalina, en donde al igual que la simulación se pueden observar fenómenos o principios que llevan a la construcción y visualización de estructuras auto organizables.

En la cristalización la aplicación de energía, se da en los cambios de temperatura de la solución y si se trata de la energía para la construcción de estructuras auto organizables sería la aplicación movimientos oscilantes y circunferenciales los que permiten la creación de dichas estructuras.

En la experimentación, se puede observar cómo nacen partículas con una forma de crecimiento naturalmente establecida, a esto se le llama auto ensamblado, que es cuando las partículas toman una forma establecida. Caso similar de auto ensamblado pasa con el juego jazz donde este se acopla a las demás piezas para dar forma al entorno que es establecido.

La partícula naciente en la cristalización, se empieza agregar en el entorno, esto se debe al cambio de temperatura dinámico y gracias a esto se empiezas a generar o a construir estructuras auto organizadas. Este es un caso muy similar en la modelización de estructuras auto organizables ya que la pieza diseñada por su forma tridimensional es capaz de agregarse una vez se le aplica dinámica al entorno que las contiene.

La simulación como proceso de modelización de estructuras auto organizables, es posible ya que la simulación permite observar de manera más simple todos los datos proyectados y experimentados en lapsos de tiempo más cortos a los reales y partiendo de estos resultados se generan análisis que permiten entender la agregación y los fenómenos que ocurren en la realidad experimentada.

Pero no es solo la simulación, sino todo el proceso que va desde la experimentación directa con la materia cristalina hasta el análisis de los modelos obtenidos de la simulación posteriormente pasando por el diseño de piezas tridimensionales que son sometidas a ejercicios en entornos dinámicos, que permitieron obtener la modelización estructuras auto organizables basados en la simulación de l agregación de partículas cristalinas.

7. Recomendaciones

Los datos que alimenten tanto la simulación como el código *Python* para la proyección, deben ser suficiente información para que los análisis generen respuestas coherentes con la realidad.

Desarrollar estructuras auto ensambladas, en base a los datos y las simulaciones derivadas de las experimentaciones de orientación de la materia cristalina realizada en las pasantías.

Bibliografía

Alvarez M. (2013). *desarrolloweb.* Obtenido de que es python: https://desarrolloweb.com/articulos/1325.php#

autoorganizacion. (23 de mayo de 2013). *wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Autoorganizaci%C3%B3n

CAID. (s.f.). *3dcadportal.* Obtenido de rhinoceros: http://www.3dcadportal.com/rhinoceros.html

Cardenas, Y. (1 de noviembre de 2010). *slideshare.* Obtenido de tipos de investigacion: https://es.slideshare.net/YACARLA/tipos-de-investigacion-5638190

Dominguez, C., Lopez, M., Martinez, G., Reyes, B., & Vazquez, M. (2008). simulacion digital. jocotitlan, mexico.

eurosur. (s.f.). Obtenido de NATURAL, CAMBIOS EN EL SISTEMA: http://www.eurosur.org/medio_ambiente/bif28.htm

Fernandez, R. (s.f.). metodologia de la modelizacion. *unidad docente de lofica y filosofia de la ciencia*.

ferrer garcia, m., & rosell , a. o. (2005). construccion y aplicacion de macromoleculas. *construccion y aplicacion de macromoleculas*. barcelona, españa.

Fullana, C., & Urqia, E. (2009). LOS MODELOS DE SIMULACIÓN:UNA HERRAMIENTA MULTIDISCIPLINAR DE INVESTIGACIÓN. madrid, españa.

Garcia, Alvaro;Ortega, Miguel. (2006). *introduccion a la simulacion de sistemas discretos*.

Jimenez, A., Anzola, j., Tarazona, G., & Bolaños, j. (2017). modelo para la simulacion de sistemas multi- agentes roboticos en python. *Udistrital*, 30-41.

Kirdeiski, G., & Vestartas, P. (2017). *AGGREGATION AND GRAPH-BASED MODELING.* Obtenido de https://gkirdeikis.wordpress.com/portfolio/aggregation-and-graph-based-modeling/

lidgi, g. (2 de 11 de 2018). *aprende todo sobre inteligencia artificial*. Obtenido de https://ligdigonzalez.com/libreria-scikit-learn-de-python/

Lidgi, G. (21 de 9 de 2018). *aprende todo sobre inteligencia artificial*. Obtenido de https://ligdigonzalez.com/introduccion-a-numpy-python-1/

Lomas, A. (2014). *Cellular Forms: an Artistic Exploration of Morphogenesis.* Obtenido de andylomas.com

Lozada, j. (2009). *investigacion aplicada: definicion, propiedad intelectual e industria.* Quito, ecuador.

Marin Diazaraque, j. m. (2014). teoria de la regresion lineal. En *estadisticas 2* (pág. capitulo 18).

Martinez Paula Andrea, S. H. (9 de 01 de 2019). *datacarpentry*. Obtenido de https://www.datacarpentry.org/python-ecology-lesson-es/

MODELAB. (2015). *mode lab*. Obtenido de http://grasshopperprimer.com/es/0-about/1-grasshopper-an-overview.html

Olson, A. (5 de agosto de 2015). *Self-assembly gets physical.* Obtenido de Nature Nanotechnology: https://www.nature.com/articles/nnano.2015.172

Perez, J. (2012). *definicion.de.* Obtenido de definicion de tendencia: https://definicion.de/tendencia/

Perez, J., & Gardey, A. (2013). *definicion.de.* Obtenido de definicion de virtual: https://definicion.de/virtual/

RODRÍGUEZ, D. (20 de 07 de 2018). *analystic lane*. Obtenido de https://www.analyticslane.com/2018/07/20/visualizacion-de-datos-con-seaborn/

Rougier Nicolas, M. M. (s.f.). *scipy lecture note* . Obtenido de https://claudiovz.github.io/scipy-lecture-notes-ES/intro/matplotlib/matplotlib.html

Scheer, D. (2014). *the death of drawing.* New York.

Tamami, C. (2015). *prezi*. Obtenido de definicion de programacion grafica: https://prezi.com/l55gwvzx0d00/definicion-de-programacion-grafica/

tarifa, e. (2001). Teoria de modelos y Simulacion. jujuy, argentina.

Tibbits, S. (2012). *selfassembly lab*. Obtenido de https://selfassemblylab.mit.edu/

Ucha, F. (2010). *DefinicionABC.* Obtenido de definicion de pronostico: https://www.definicionabc.com/ciencia/pronostico.php

wikipedia. (24 de octubre de 2019). Obtenido de anaconda: https://www.anaconda.com/what-is-anaconda/ ; https://es.wikipedia.org/wiki/Anaconda_(distribuci%C3%B3n_de_Python)

Anexos

Anexo A Cronograma de actividades

mes.2019-2020	1				2				3				4				5 (2020)				6				7			
actividad/ semana	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. revisión, estado de la cuestión																												
2. estudio y proyección de las tendencias de crecimiento de la materia en la simulación.																												
3.simulación y estudio de simulación (grasshopper, python)																												
4. comparación de resultados de la simulación y la experimentación con material cristalino																												
5. diseño y modelado de estructuras auto organizables																												
6.materialización de estructuras auto organizables																												
7.sistematización.																												

hace referencia a la materialización (impresión 3d y pruebas) que no se pudo realizar por la pandemia covid- 19

Anexo B Capturas

[30]:
```
matplotlib inlin
impo nu
impo pand
impo seabor s
f matplotlib impo pypl p
```

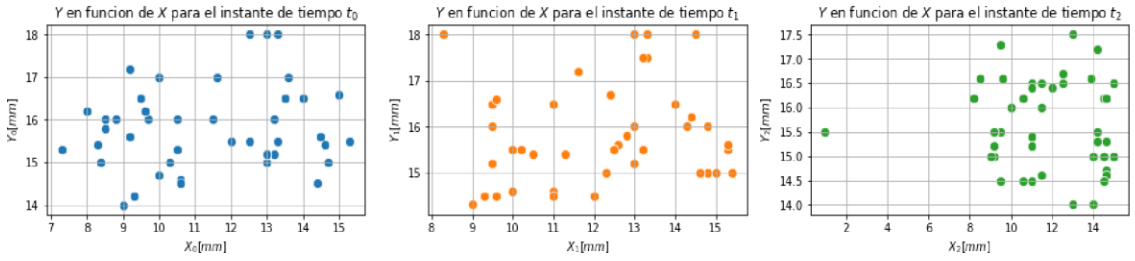
[31]:
```
d2df = pd.read_excel(r'C:\Users\ufps\Desktop\pasantia\SEMANA3\eduardo\DATOS EXCE
d2df.drop(['Z_0', 'Z_1', 'Z_2'], axis=1, inplace=True)
d2df
```

Out[31]:

	X_0	Y_0	X_1	Y_1	X_2	Y_2
0	9.2	15.6	14.3	16.0	9.5	15.5
1	8.4	15.0	13.2	15.5	9.2	15.5
2	8.8	16.0	13.0	16.0	9.2	15.5
3	8.5	16.0	12.8	15.8	9.2	15.2
4	8.5	16.0	12.6	15.6	9.0	15.0
5	8.5	16.0	12.5	15.5	9.5	14.5
6	8.5	16.0	12.5	15.5	9.2	15.0
7	8.5	16.0	12.5	15.5	9.0	15.0
8	8.5	16.0	12.5	15.5	9.0	15.0
9	8.5	15.8	9.3	14.5	9.6	16.6
10	9.3	14.2	10.0	14.6	11.0	14.5
11	8.3	15.4	9.0	14.3	9.5	17.3
12	8.0	16.2	8.3	18.0	8.5	16.6
13	7.3	15.3	8.3	18.0	8.5	16.6
14	14.5	15.6	9.5	16.0	14.5	16.2
15	13.2	16.0	10.0	15.5	14.2	15.5
16	13.3	15.5	10.2	15.5	14.6	15.3
17	13.2	15.2	10.5	15.4	14.2	15.3
18	13.0	15.0	9.5	15.2	14.0	15.0
19	13.0	15.0	11.0	14.6	12.5	16.5
20	13.0	15.2	9.5	16.5	13.0	17.5
21	13.0	15.2	9.5	16.5	13.0	17.5
22	13.0	15.2	9.6	16.6	13.0	17.5
23	10.6	14.6	11.3	15.4	11.5	16.5
24	9.2	17.2	11.6	17.2	8.2	16.2
25	10.0	14.7	11.0	14.5	11.5	14.6
26	9.0	14.0	9.6	14.5	10.6	14.5
27	9.0	14.0	9.6	14.5	11.5	16.0
28	9.7	16.0	14.8	16.0	10.0	16.0
29	9.6	16.2	14.4	16.2	14.6	16.2
30	10.5	16.0	15.3	15.5	10.6	16.2
31	9.5	16.5	14.8	15.0	11.0	15.2
32	9.5	16.5	15.0	15.0	11.0	16.5

X_0	Y_0	X_1	Y_1	X_2	Y_2
10.0	17.0	13.0	18.0	11.0	16.4
10.5	15.3	14.0	16.5	11.0	16.4
10.5	15.3	13.3	18.0	11.0	15.4
10.3	15.0	13.0	18.0	13.0	17.5
12.5	15.5	13.0	15.2	13.0	14.0
12.0	15.5	12.3	15.0	12.0	16.4
11.6	17.0	12.3	15.0	14.0	14.0
11.5	16.0	12.0	14.5	14.0	14.0
10.6	14.5	12.0	14.5	14.0	14.0
14.6	15.4	15.3	15.6	14.2	17.2
13.6	17.0	13.3	17.5	13.9	16.6
15.0	16.6	13.2	17.5	15.0	16.5
15.3	15.5	11.0	16.5	14.5	15.0
13.5	16.5	11.3	15.4	15.0	15.0
13.3	18.0	14.6	15.0	15.0	15.0
14.0	16.5	14.6	15.0	14.6	14.7
14.0	16.5	11.0	16.5	14.6	14.6
13.0	18.0	14.0	16.5	12.5	16.7
14.4	14.5	15.4	15.0	14.5	14.5
12.5	18.0	12.4	16.7	1.0	15.5
14.7	15.0	14.5	18.0	1.0	15.5

[32]:
```
plt.figure(figsize=(18,3))
plt.subplot(131)
ax = sns.scatterplot(d2df['X_0'], d2df['Y_0'],
                     color=sns.color_palette("tab10", n_colors=10)[0],
                     s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

plt.subplot(132)
ax1 = sns.scatterplot(d2df['X_1'], d2df['Y_1'],
                      color=sns.color_palette("tab10", n_colors=10)[1],
                      s=70)
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_1$')
plt.grid()

plt.subplot(133)
ax2 = sns.scatterplot(d2df['X_2'],d2df['Y_2'],
                      color=sns.color_palette("tab10", n_colors=10)[2],
                      s=70)
plt.xlabel(r'$X_2[mm]$')
plt.ylabel(r'$Y_2[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_2$')
plt.grid()
```

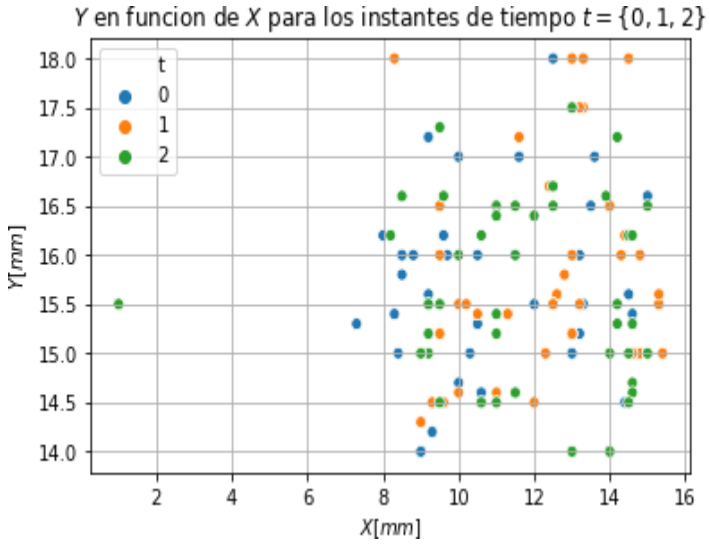


```
[34]: d2df['t_0'] = np.zeros(len(d2df)) d2df['t_1'] = np.ones(len(d2df)) d2df['t_2'] = np.ones(len(d2df))*2
seed2 = np.vstack((d2df[['X_0', 'Y_0', 't_0']].values,                d2df[['X_1', 'Y_1', 't_1']].values,
d2df[['X_2', 'Y_2', 't_2']].values)) seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 't']) seed2_df['t'] =
seed2_df['t'].astype(int) print('Dimension: ', seed2_df.shape) seed2_df.sample(10)
Dimension:  (162, 3)
```

Out[34]:

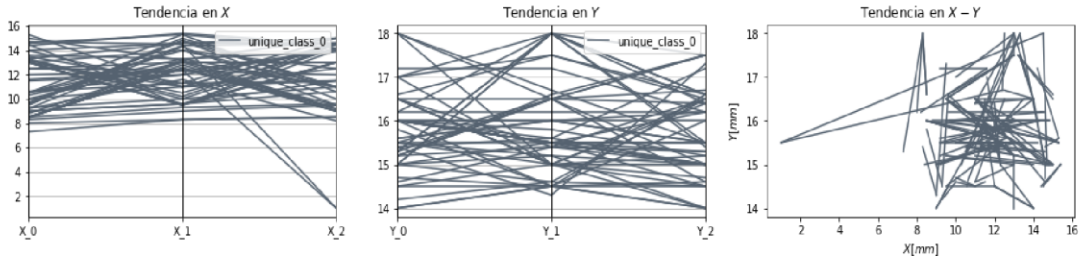
	x	Y	t
121	8.5	16.6	2
85	14.8	15.0	1
150	14.2	17.2	2
158	12.5	16.7	2
157	14.6	14.6	2
151	13.9	16.6	2
129	13.0	17.5	2
93	12.3	15.0	1
76	9.6	16.6	1
36	10.3	15.0	0

```
[35]:  s  scatterplot  seed2_d  '  seed2_d  '  h  seed2_d 't
                                palette  s  color_palette  "tab10  n_color
    s  set_palette  "tab10
    p  xlabe  r'$X[mm]$
    p  ylabe  r'$Y[mm]$
    p  title  r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$'
    p  gr
```




```
[36]:  delta1  seed2_d  l  seed2_d 't  '  value  seed2_d  l  seed2_d 't
       delta2  seed2_d  l  seed2_d 't  '  value  seed2_d  l  seed2_d 't
```

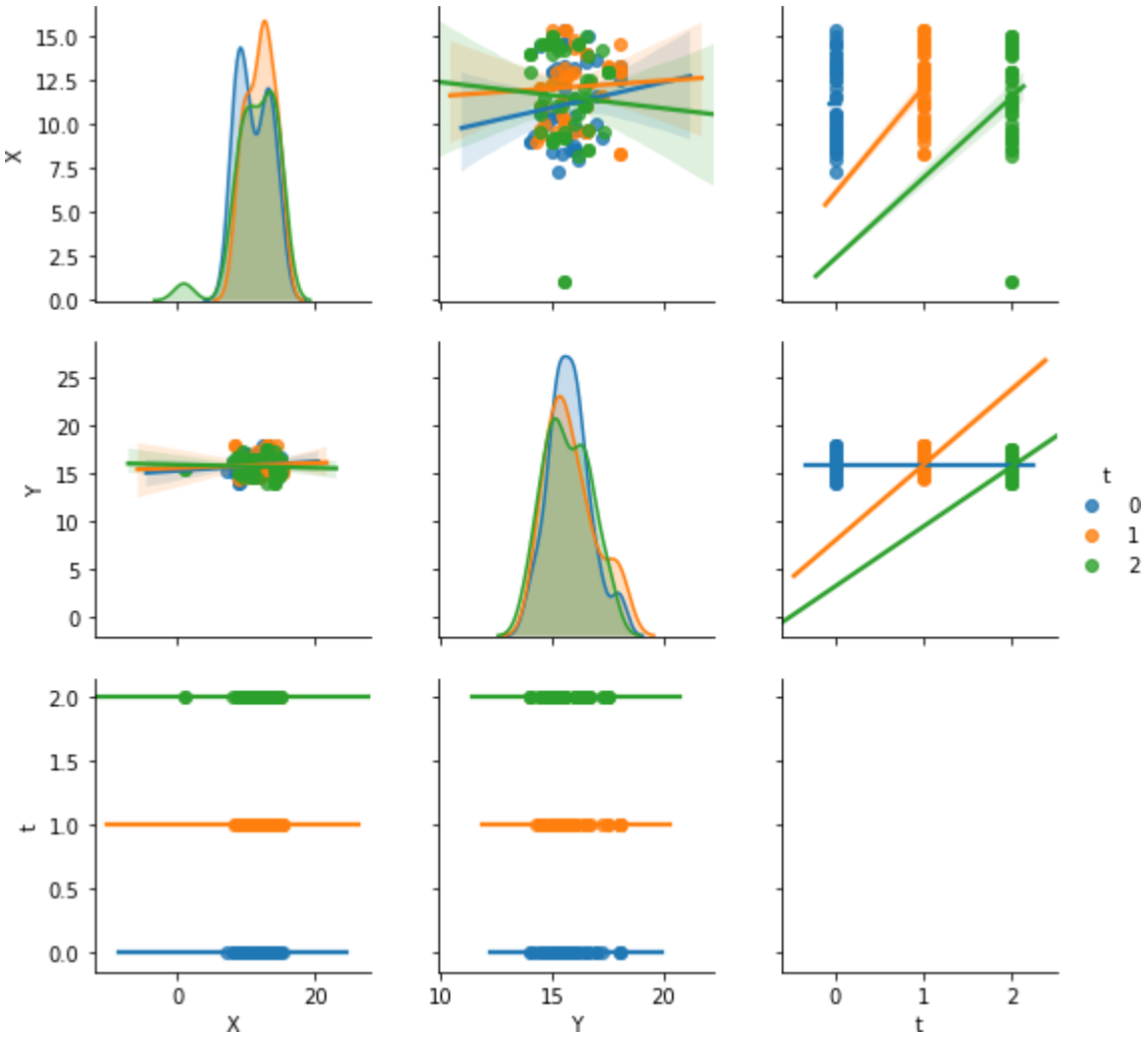
[37]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
for i in range(len(df_0)):
    plt.plot([df_0.loc[i, 'X'], df_1.loc[i, 'X']],
             [df_0.loc[i, 'Y'], df_1.loc[i, 'Y']],
             [df_1.loc[i, 'X'], df_2.loc[i, 'X']],
             [df_1.loc[i, 'Y'], df_2.loc[i, 'Y']], color=('#556270'))
plt.xlabel(r'$X[mm]$')
plt.ylabel(r'$Y[mm]$')
plt.title(r'Tendencia en $X-Y$')
plt.show()
```



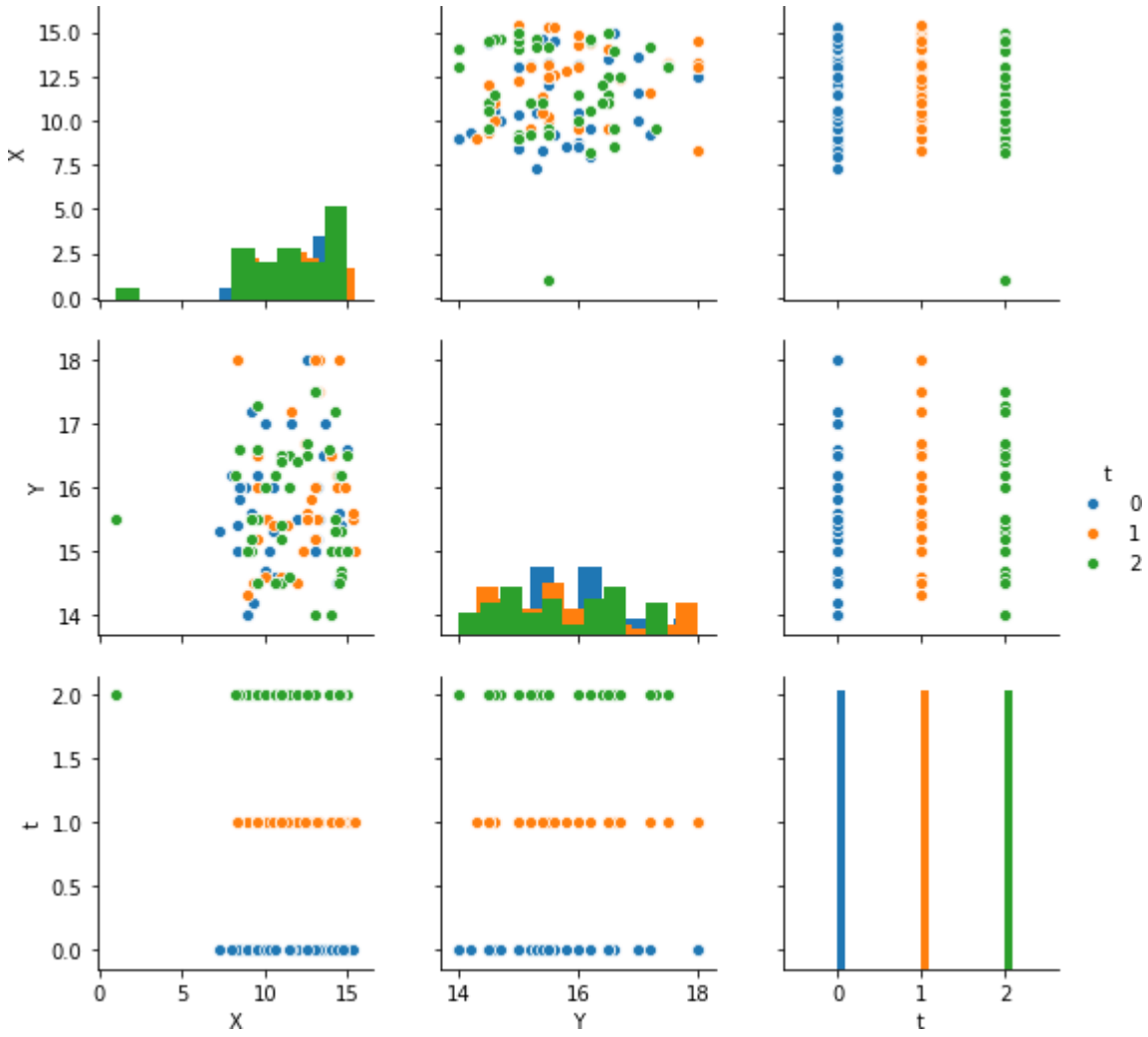
[38] :hue='t', kind="reg")

C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

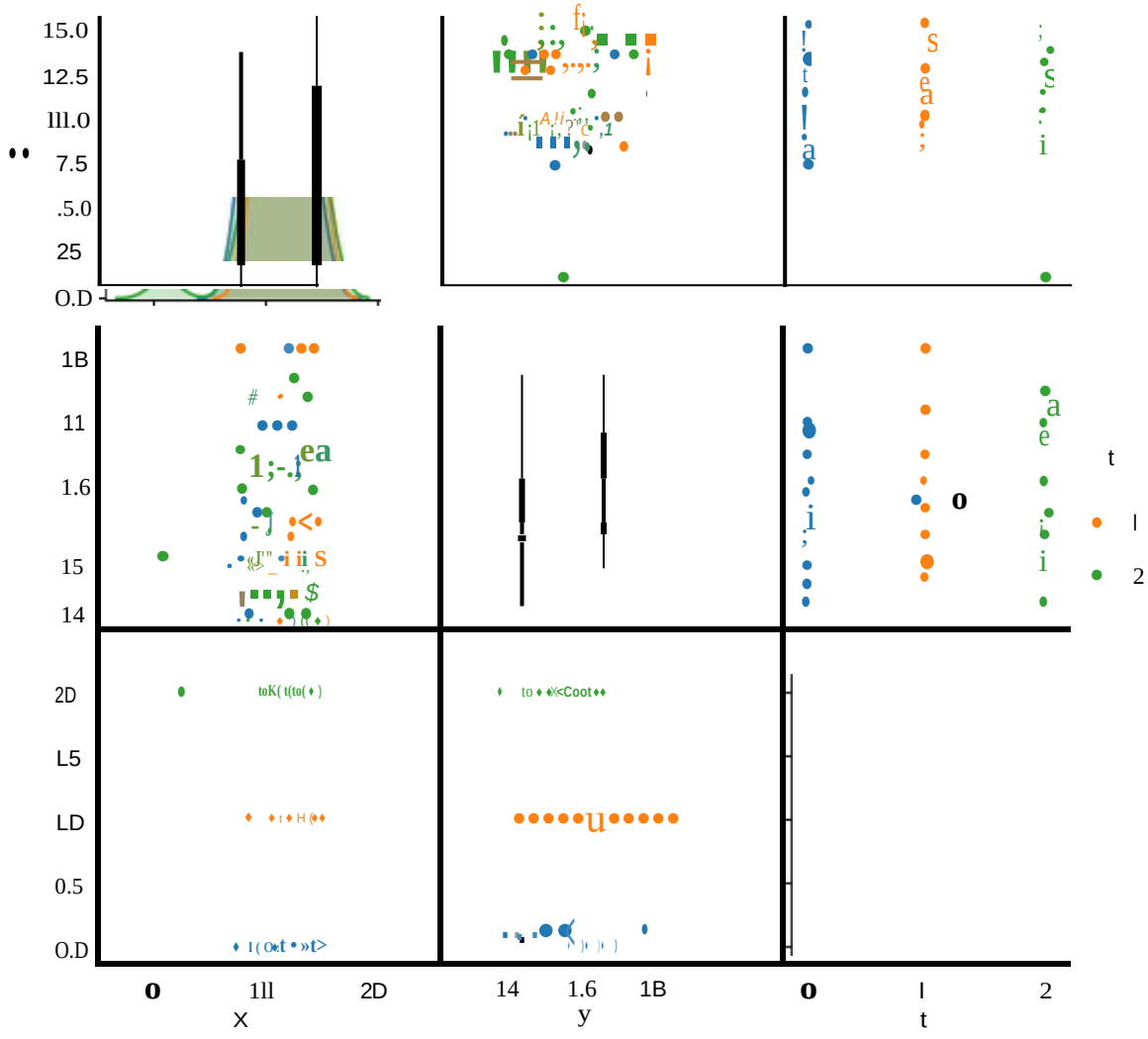
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



[39] :hue='t', diag_kind='hist')

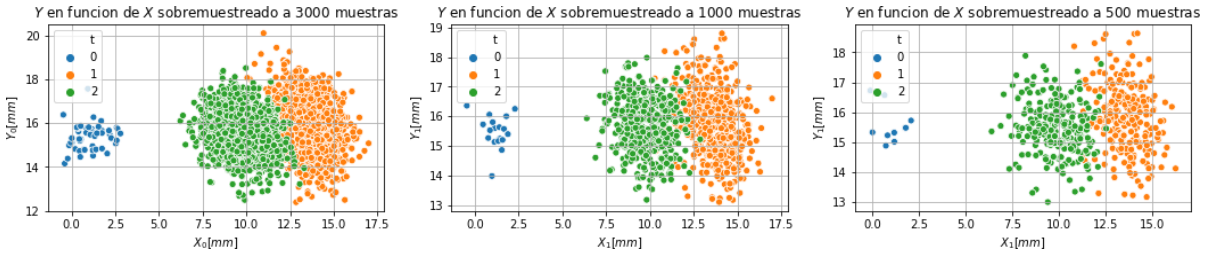

X
Y
t
15.0
12.5
10.0
7.5
5.0
2.5
0.0
18
17
16
15
14
2.0
1.5
1.0
0.5
0.0
0
5
10
15
14
16
18
0
1
2
t
0
1
2

[40]:hue='t')


15.0
12.5
lll.0
7.5
.5.0
25
O.D
1B
11
1.6
15
14
2D
L5
LD
0.5
O.D
o
1ll
2D
14
1.6
1B
o
l
2
X
y
t
t
l
2

[41] :
```
from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(2),
init_params="kmeans", max_iter=100, random_state=2).fit(seed2_df[['X', 'Y']])
```

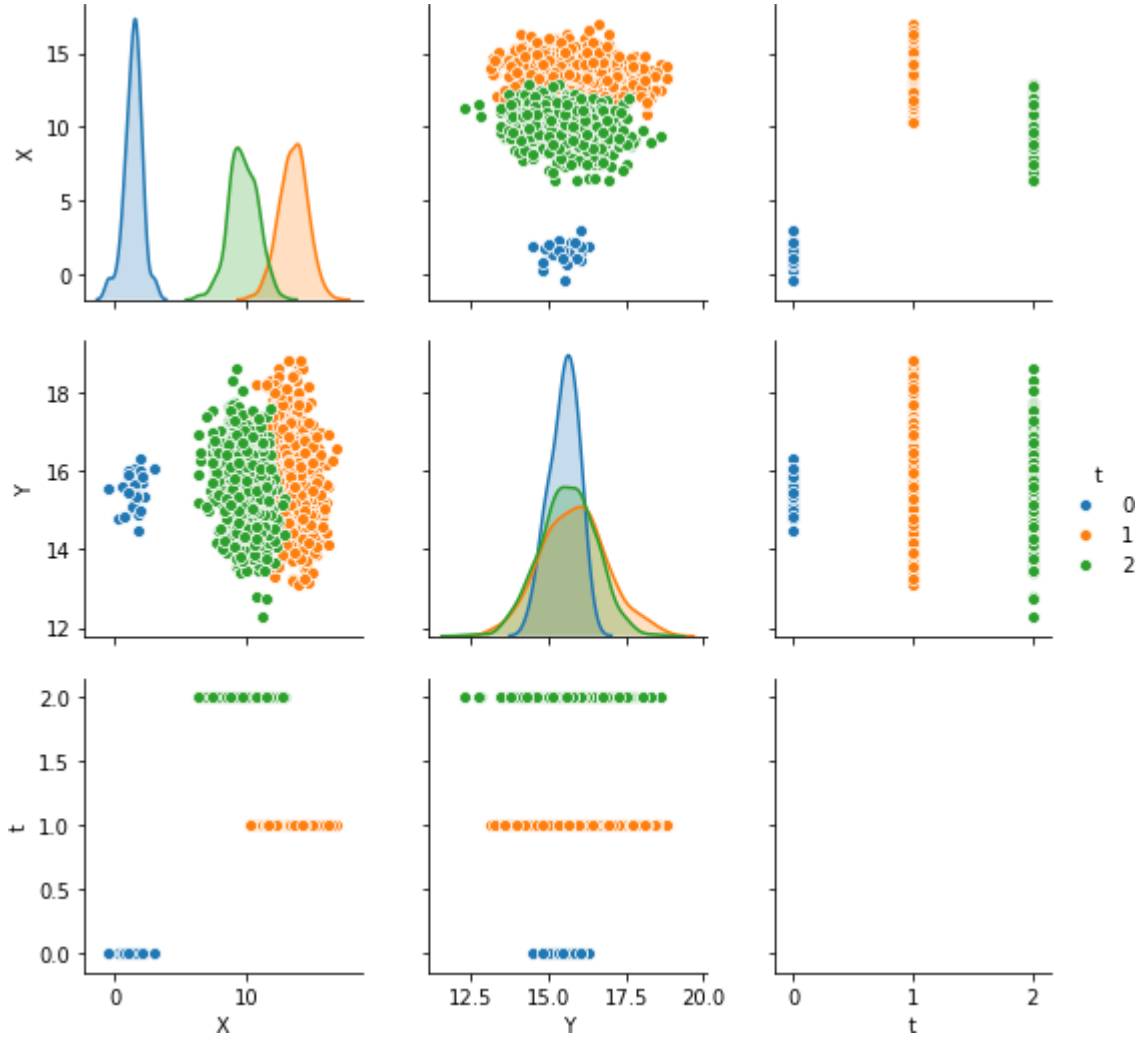
[42] :
```
plt.figure(figsize=(18,3)) plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('Jorge_2d_3000_muestras.csv') ax = sns.scatterplot(seed2over_df['X'],
seed2over_df['Y'],                hue=seed2over_df['t'],
palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$') plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$') plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```



[43] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']), pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=**True**)
sns.pairplot(seed2over_df, hue='t') plt.show()

C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



Anexo C Análisis

Out[6]:

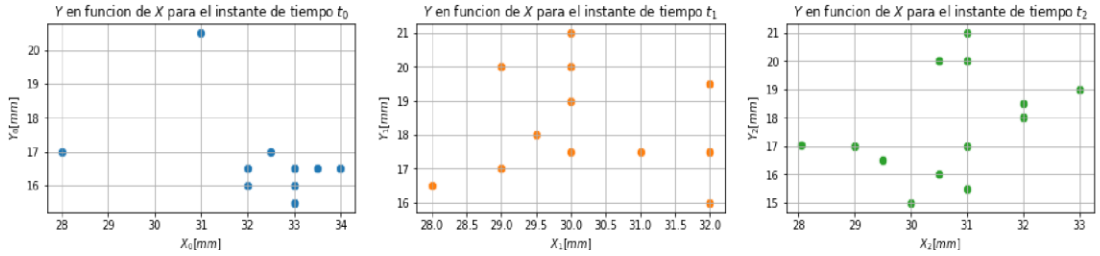
[1]:
```
  matplotlib   inline
impo   num
impo   panda
impo   seaborn      s
fr   matplotlib   impo   pyplot      p
```

[6]:
```
d2          read_excel   r'C:\Users\ufps\Desktop\pasantia\SEMANA3\jhon\jhon2d.xlsx'
d2   dro    'Z_0'    'Z_1'    'Z_2'      axi        inplace   Tr
d2
```

	X	Y	X	Y	X	Y
	3	1	3	1	3: .	15.
1	16.5	32.0	16.0	30.50	16.00	33.
5						
2	33.16.5	31.0	17.5	30.00	15.00	
0						
3	32.17.0	30.0	17.5	29.50	16.50	
5						
4	34.16.5	29.5	18.0	29.00	17.00	
0						
5	34.16.5	29.5	18.0	28.05	17.05	
0						
6	33.15.5	28.0	16.5	31.00	17.00	
0						
7	33.16.0	29.0	17.0	32.00	18.00	
0						
8	32.16.0	29.5	18.0	33.00	19.00	
0						
9	32.16.5	29.0	20.0	32.00	18.00	
0						

10						
	33.15.5	30.0	21.0	32.00	18.50	
0						
11						
	33.15.5	30.0	20.0	31.00	21.00	
0						
12						
	28.17.0	30.0	19.0	30.50	20.00	
0						
13						
	31.					
0	20.5	32.0	19.5	31.00	20.00	

In [7]:
```
plt.figure(figsize=(18,3))
plt.subplot(131)
ax = sns.scatterplot(d2df['X_0'], d2df['Y_0'],
                     color=sns.color_palette("tab10", n_colors=10)[0],
                     s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

plt.subplot(132)
ax1 = sns.scatterplot(d2df['X_1'], d2df['Y_1'],
                      color=sns.color_palette("tab10", n_colors=10)[1],
                      s=70)
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_1$')
plt.grid()

plt.subplot(133)
ax2 = sns.scatterplot(d2df['X_2'],d2df['Y_2'],
                      color=sns.color_palette("tab10", n_colors=10)[2],
                      s=70)
plt.xlabel(r'$X_2[mm]$')
plt.ylabel(r'$Y_2[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_2$')
plt.grid()
```

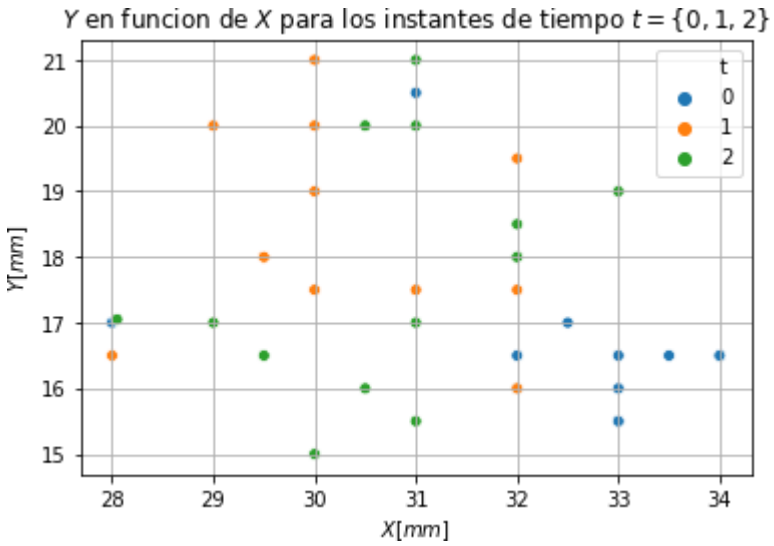


In [9]: d2df['t_0'] = np.zeros(len(d2df)) d2df['t_1'] = np.ones(len(d2df)) d2df['t_2'] = np.ones(len(d2df))*2
seed2 = np.vstack((d2df[['X_0', 'Y_0', 't_0']].values, d2df[['X_1', 'Y_1', 't_1']].values, d2df[['X_2', 'Y_2', 't_2']].values)) seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 't']) seed2_df['t'] = seed2_df['t'].astype(int) print('Dimension: ', seed2_df.shape) seed2_df.sample(20)
Dimension: (42, 3)

Out[9]: x **Y** **t**

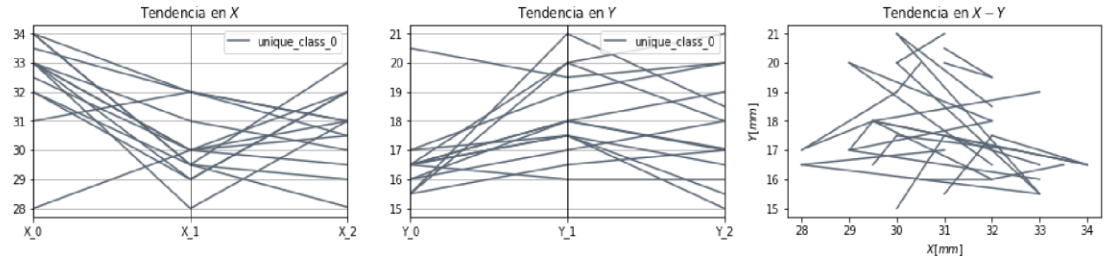
	25	30.0	20.0	1
	36	33.0	19.0	2
	24	30.0	21.0	1
	27	32.0	19.5	1
32.0	17.5	1		
32.0	16.0	1		
	1	33.5	16.5	0
	37	32.0	18.0	2
	0	34.0	16.5	0
	6	33.0	15.5	0
	21	29.0	17.0	1
	29	30.5	16.0	2
	11	33.0	15.5	0
	34	31.0	17.0	2
	41	31.0	20.0	2
	35	32.0	18.0	2
	7	33.0	16.0	0
	5	34.0	16.5	0
	12	28.0	17.0	0
	2	33.0	16.5	0

```
[10]:          s    scatterplot    seed2_df '         seed2_df '         h    seed2_df 't'
                                                palette    s    color_palette    "tab10"    n_colors
          s    set_palette    "tab10"
          p    xlabel    r'$X[mm]$'
          p    ylabel    r'$Y[mm]$'
          p    title    r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$'
          p    gri
```



In [11]: delta10 = seed2_df.loc[seed2_df['t'] == 1]['Y'].values-seed2_df.loc[seed2_df['t' delta21 = seed2_df.loc[seed2_df['t'] == 2]['Y'].values-seed2_df.loc[seed2_df['t'

In [12]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
for i in range(len(df_0)):
    plt.plot([df_0.loc[i, 'X'], df_1.loc[i, 'X']],
             [df_0.loc[i, 'Y'], df_1.loc[i, 'Y']],
             [df_1.loc[i, 'X'], df_2.loc[i, 'X']],
             [df_1.loc[i, 'Y'], df_2.loc[i, 'Y']], color=('#556270'))
plt.xlabel(r'$X[mm]$')
plt.ylabel(r'$Y[mm]$')
plt.title(r'Tendencia en $X-Y$')
plt.show()
```

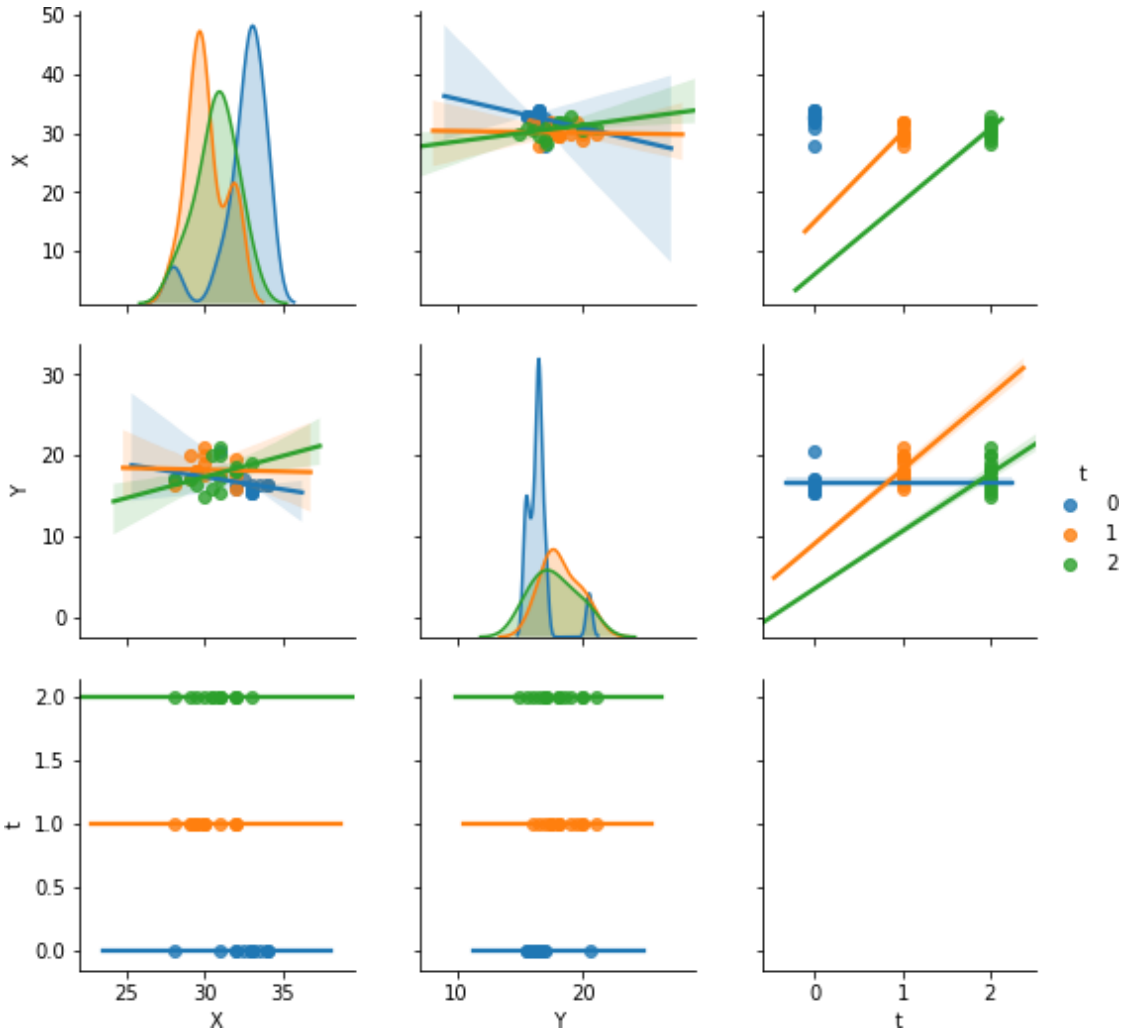


C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:

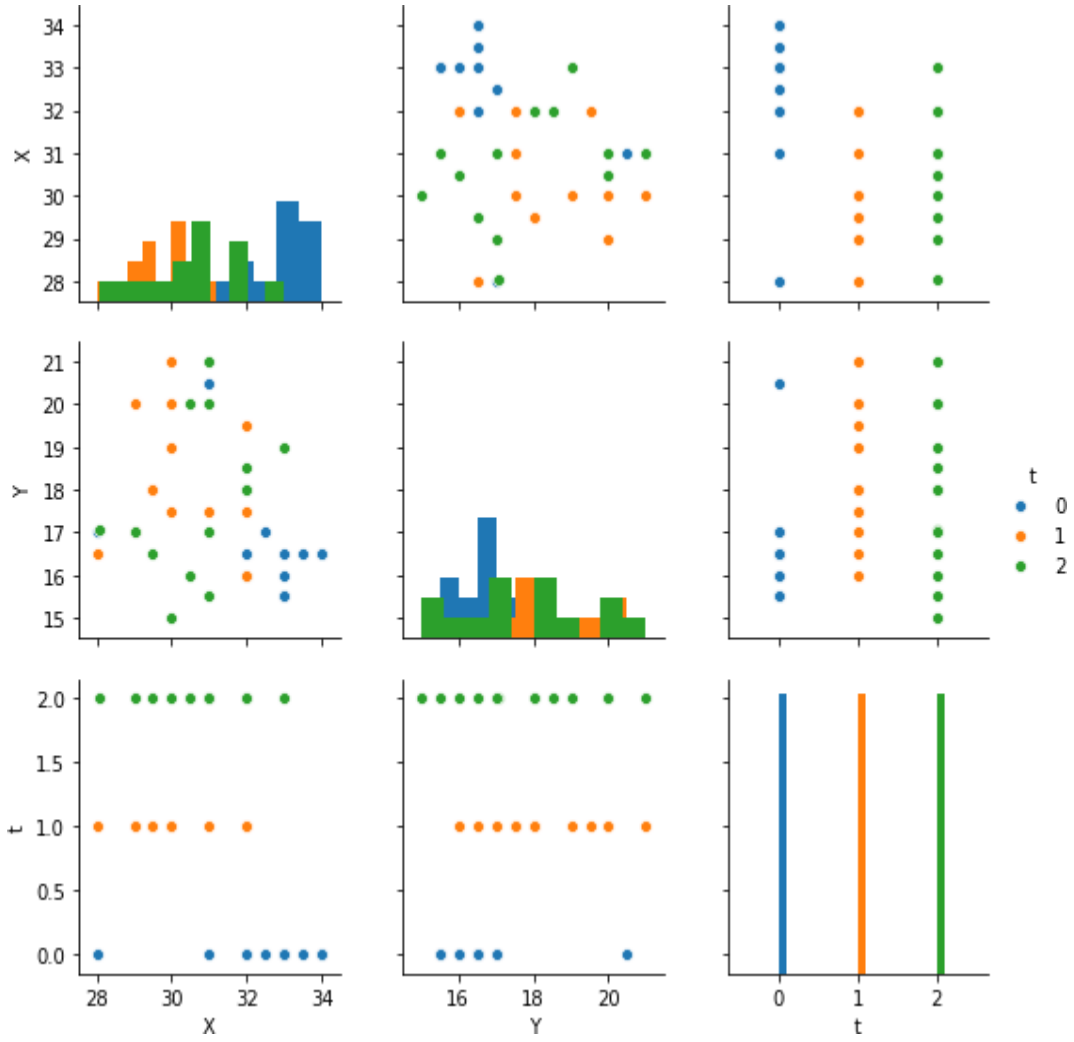
[13]:
```
s    pairplot    seed2_df    h    't'    kin    "reg
p    sh
```

RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

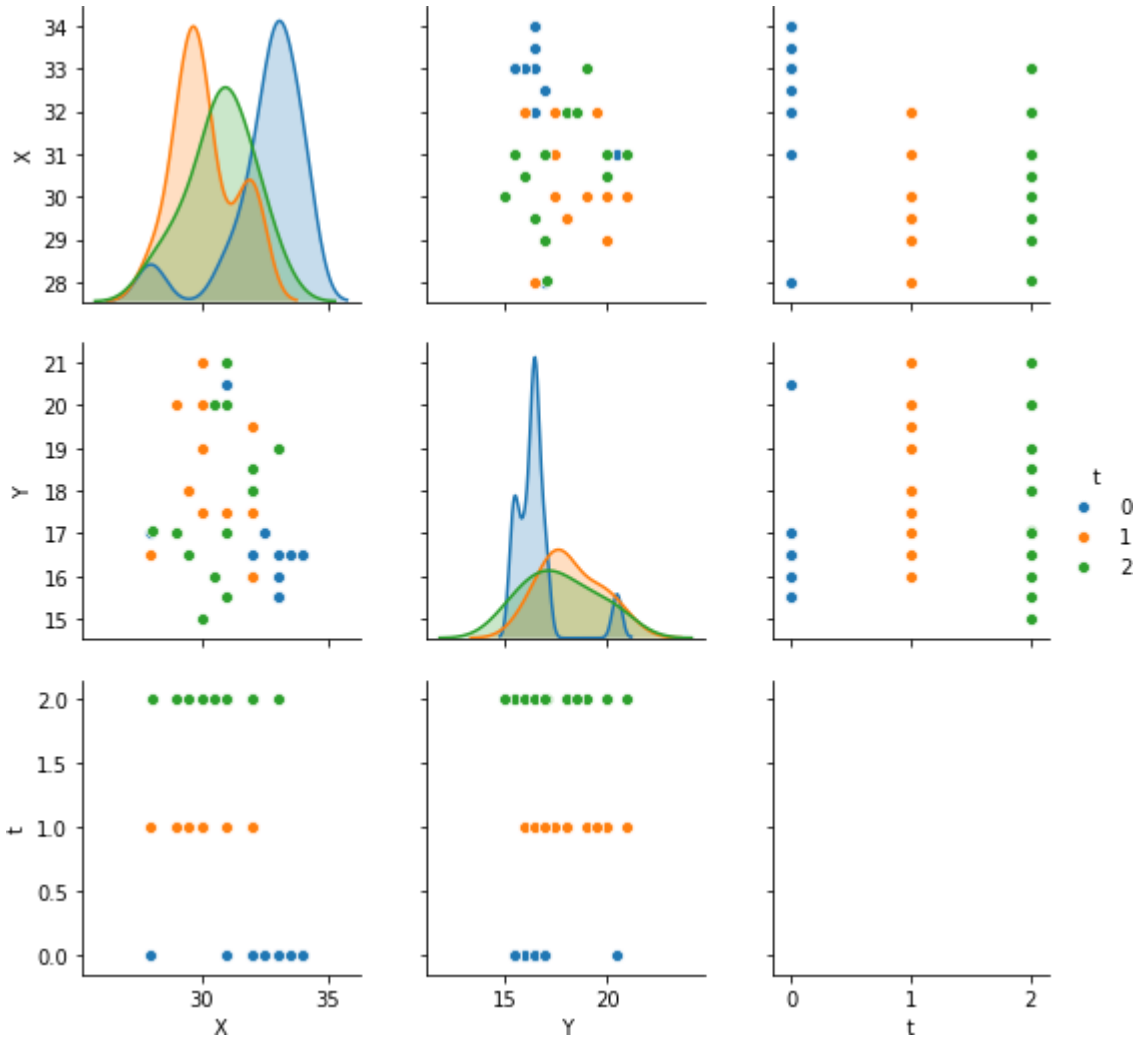
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2


X
Y
t
0
1
2

[14]:
```
sns.pairplot(seed2_df, hue='t', diag_kind='hist')
plt.show()
```

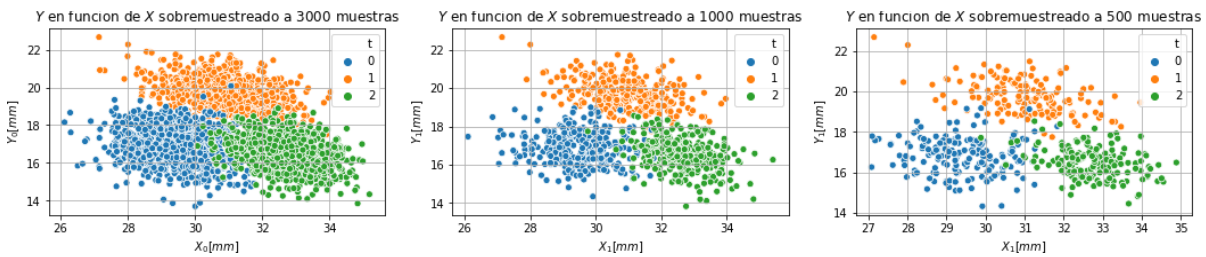


[15]:
```
s    pairplot    seed2_df    h    't'
p    sh
```

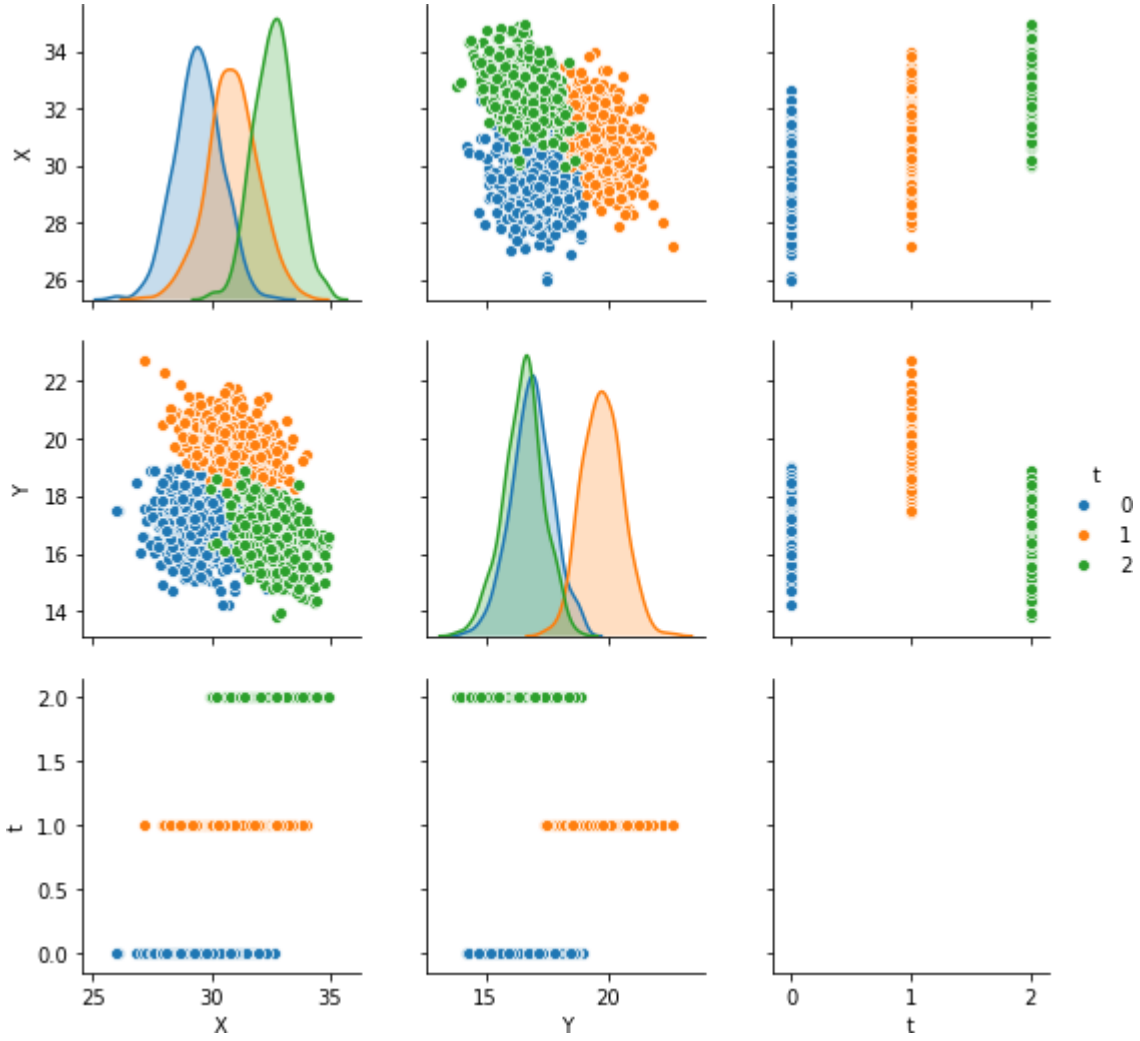


```
[16] : from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
   n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(2),
   init_params="kmeans", max_iter=100, random_state=2).fit(seed2_df[['X', 'Y']])
```

```
[17] : plt.figure(figsize=(18,3)) plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('JHON_2d_3000_muestras.csv') ax = sns.scatterplot(seed2over_df['X'],
seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```



```
[18] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s,
columns=['X', 'Y']),                    pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
sns.pairplot(seed2over_df, hue='t') plt.show()
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:
RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta
* nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning:
invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2
```



Anexo D Análisis

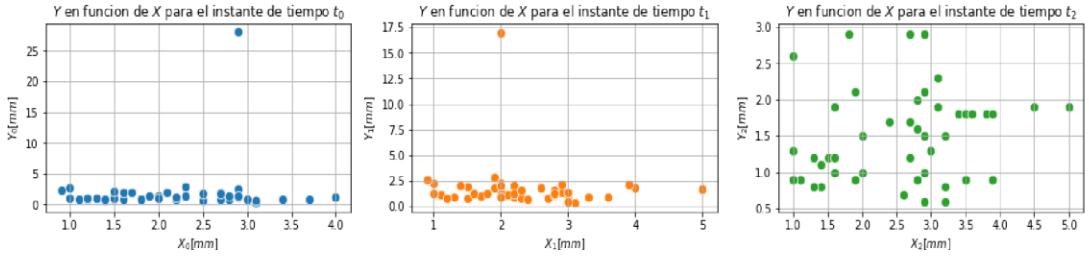
```
[1]:  matplotlib  inline
      impo  num
      impo  panda
      impo  seaborn  s
      fr  matplotlib  impo  pyplot  p
```

```
In [2]: d2df = pd.read_excel(r'C:\Users\ufps\Desktop\pasantia\SEMANA3\jorge\jorge 2d.xls
        d2df.drop(['Z_0', 'Z_1', 'Z_2'], axis=1, inplace=True)
        d2df
```

	X_0	Y_0	X_1	Y_1	X_2	Y_2
0	1.5	1.0	2.2	1.0	1.6	1.0
1	1.2	0.9	2.2	0.9	1.4	1.1
2	1.3	1.0	1.0	2.3	2.0	1.0
3	1.0	1.0	3.0	0.9	1.8	2.9
4	2.0	1.5	2.2	2.0	1.9	0.9
5	1.8	0.8	2.1	1.1	1.0	1.3
6	1.1	0.9	2.3	0.8	1.5	1.2
7	1.4	0.7	2.4	0.7	1.4	0.8
8	1.5	0.9	1.5	0.8	1.1	0.9
9	1.2	0.8	2.7	0.8	1.6	1.2
10	1.5	1.1	2.8	1.1	2.9	1.0
11	2.0	1.0	2.0	1.0	1.0	2.6
12	0.9	2.3	1.7	1.0	3.9	0.9
13	3.0	0.8	2.0	1.3	3.1	2.3
14	2.7	2.0	1.9	1.8	3.1	1.9
15	2.3	1.4	1.5	1.9	2.8	0.9
16	2.7	0.9	1.6	1.2	2.6	0.7
17	2.5	0.7	1.3	0.9	1.3	0.8
18	1.4	0.8	1.1	1.1	2.9	0.6
19	2.8	0.8	3.0	1.0	3.0	1.3
20	2.9	1.2	0.9	2.6	2.0	1.5
21	1.6	0.8	3.1	0.3	5.0	1.9
22	2.1	1.9	4.0	1.8	1.9	2.1
23	1.7	2.0	3.0	0.5	1.0	0.9
24	1.6	2.0	2.0	0.9	1.3	1.2
25	1.9	1.5	1.2	0.8	2.7	2.9
26	1.2	0.9	3.0	0.5	2.9	2.1
27	1.2	1.1	2.9	1.4	4.5	1.9
28	2.3	2.9	2.0	17.0	3.2	0.6
29	1.0	2.8	5.0	1.7	2.9	1.5
30	4.0	1.3	1.9	2.8	3.0	1.3
31	3.1	0.2	1.0	1.2	3.2	0.8
32	2.2	0.9	1.8	1.2	2.7	1.7

X_0	Y_0	X_1	Y_1	X_2	Y_2
33	0.9	2.9	2.1	2.8	2.0
1	1.				
34					
1	3. 0.6	2.0	2.3	1.6	1.9
35					
8	2. 1.5	3.9	2.2	2.9	2.9
36					
9	2. 1.2	3.3	0.9	3.5	0.9
37					
9	1. 1.5	2.8	1.6	2.4	1.7
38					
9	2. 2.5	3.0	1.4	2.8	1.6
39					
9	2. 28.0	2.2	1.2	2.7	1.2
40					
4	3. 0.9	2.6	1.8	2.4	1.7
41					
7	2. 1.7	2.0	2.0	3.4	1.8
42					
9	2. 1.5	1.4	2.0	3.2	1.5
43					
2	2. 1.3	3.6	0.9	3.5	1.8
44					
5	2. 1.8	2.3	1.6	3.6	1.8
45					
5	1. 2.1	2.8	1.2	3.8	1.8
46					
7	3. 0.9	2.3	1.6	3.9	1.8

In [3]:
```
plt.figure(figsize=(18,3))
plt.subplot(131)
ax = sns.scatterplot(d2df['X_0'], d2df['Y_0'],
                     color=sns.color_palette("tab10", n_colors=10)[0],
                     s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

plt.subplot(132)
ax1 = sns.scatterplot(d2df['X_1'], d2df['Y_1'],
                      color=sns.color_palette("tab10", n_colors=10)[1],
                      s=70)
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_1$')
plt.grid()

plt.subplot(133)
ax2 = sns.scatterplot(d2df['X_2'],d2df['Y_2'],
                      color=sns.color_palette("tab10", n_colors=10)[2],
                      s=70)
plt.xlabel(r'$X_2[mm]$')
plt.ylabel(r'$Y_2[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_2$')
plt.grid()
```

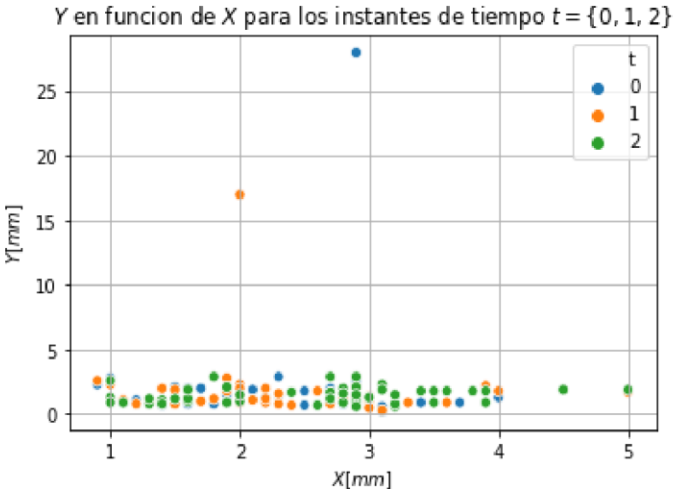


In [6]:
```
d2df['t_0'] = np.zeros(len(d2df))
d2df['t_1'] = np.ones(len(d2df))
d2df['t_2'] = np.ones(len(d2df))*2
seed2 = np.vstack((d2df[['X_0', 'Y_0', 't_0']].values,
                   d2df[['X_1', 'Y_1', 't_1']].values,
                   d2df[['X_2', 'Y_2', 't_2']].values))
seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 't'])
seed2_df['t'] = seed2_df['t'].astype(int)
print('Dimension: ', seed2_df.shape)
seed2_df.sample(20)
```

```
Dimension:  (141, 3)
```

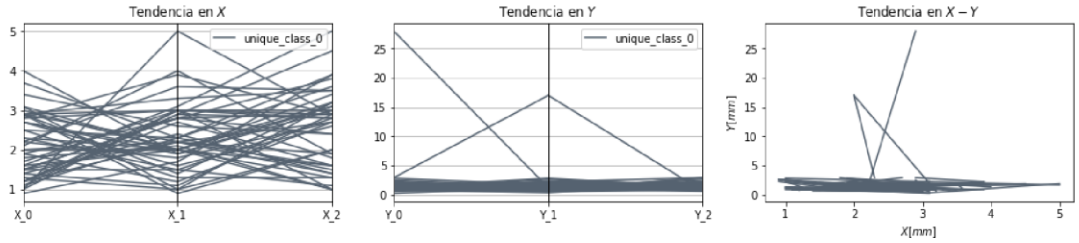
	X	Y	t
50	3.0	0.9	1
137	3.5	1.8	2
17	2.5	0.7	0
20	2.9	1.2	0
109	2.8	0.9	2
39	2.9	28.0	0
25	1.9	1.5	0
68	3.1	0.3	1
110	2.6	0.7	2
64	1.3	0.9	1
80	2.9	2.1	1

In [14]:
```
ax = sns.scatterplot(seed2_df['X'], seed2_df['Y'], hue=seed2_df['t'],
                     palette=sns.color_palette("tab10", n_colors=3))
sns.set_palette("tab10")
plt.xlabel(r'$X[mm]$')
plt.ylabel(r'$Y[mm]$')
plt.title(r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$')
plt.grid()
```



[15]: delta10 = seed2_df.loc[seed2_df['t'] == 1]['Y'].values-seed2_df.loc[seed2_df['t' delta21 = seed2_df.loc[seed2_df['t'] == 2]['Y'].values-seed2_df.loc[seed2_df['t'

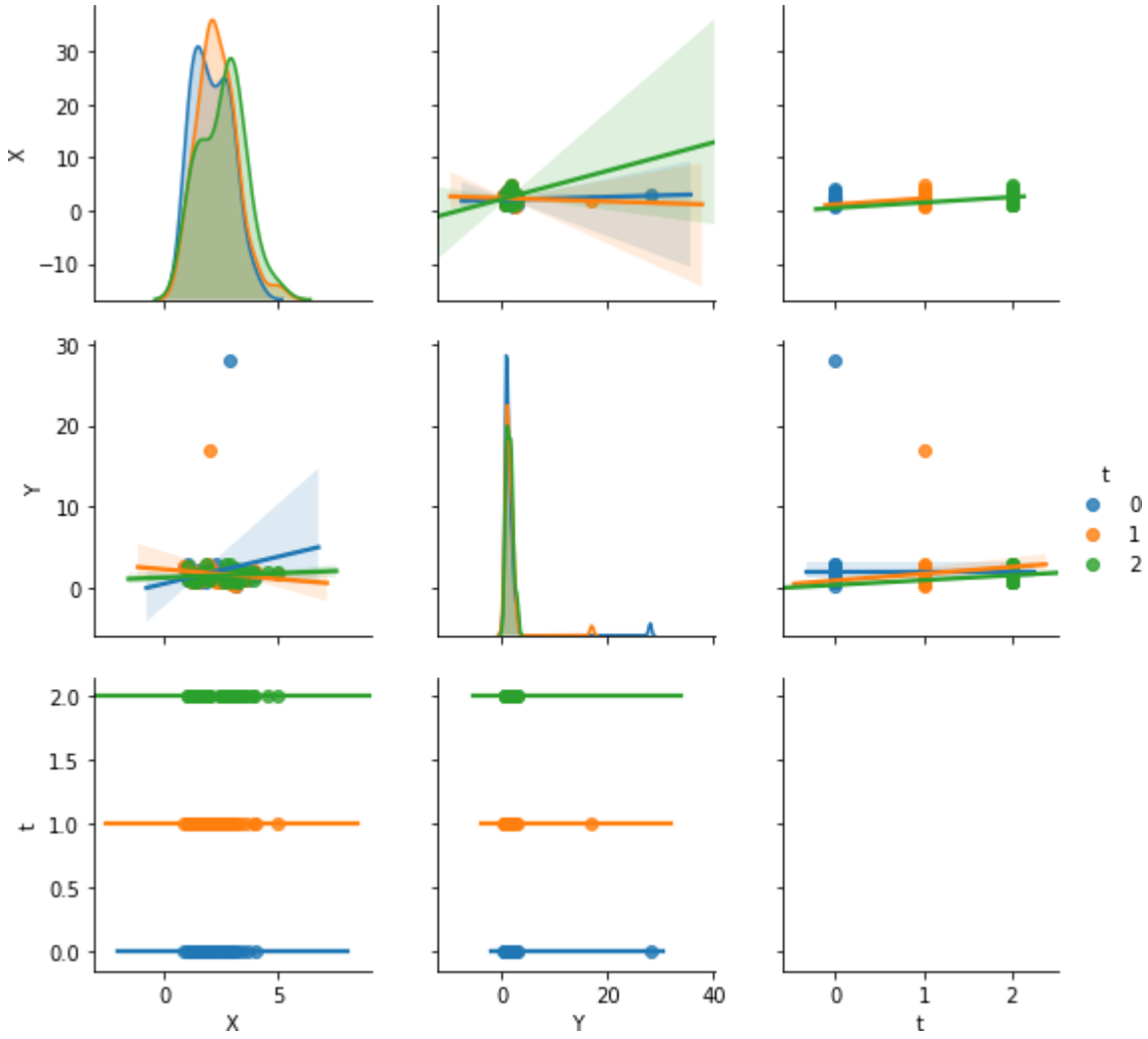
In [16]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
for i in range(len(df_0)):
    plt.plot([df_0.loc[i, 'X'], df_1.loc[i, 'X']],
             [df_0.loc[i, 'Y'], df_1.loc[i, 'Y']],
             [df_1.loc[i, 'X'], df_2.loc[i, 'X']],
             [df_1.loc[i, 'Y'], df_2.loc[i, 'Y']], color=('#556270'))
plt.xlabel(r'$X[mm]$')
plt.ylabel(r'$Y[mm]$')
plt.title(r'Tendencia en $X-Y$')
plt.show()
```



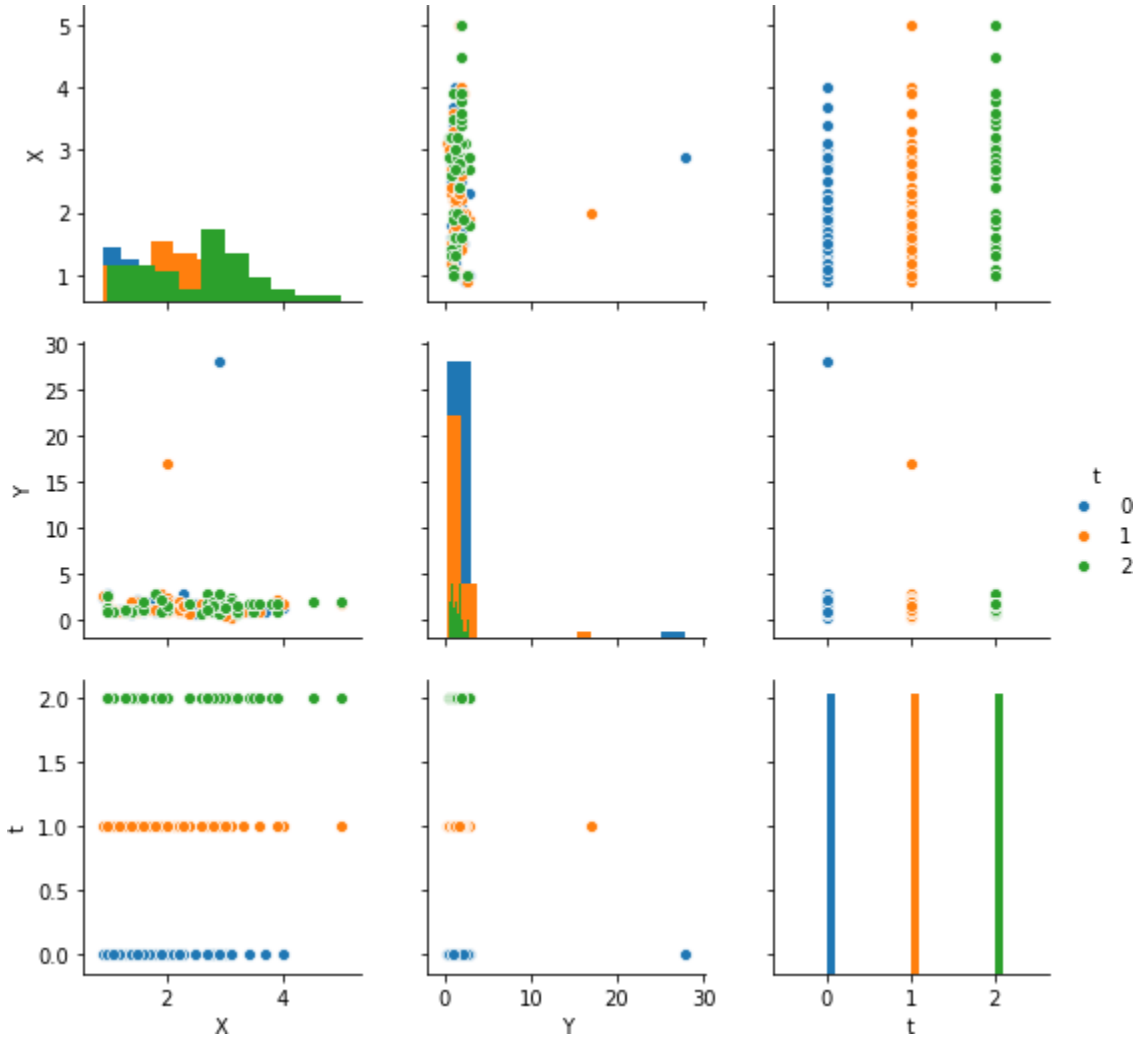
[17] :hue='t', kind="reg")

C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

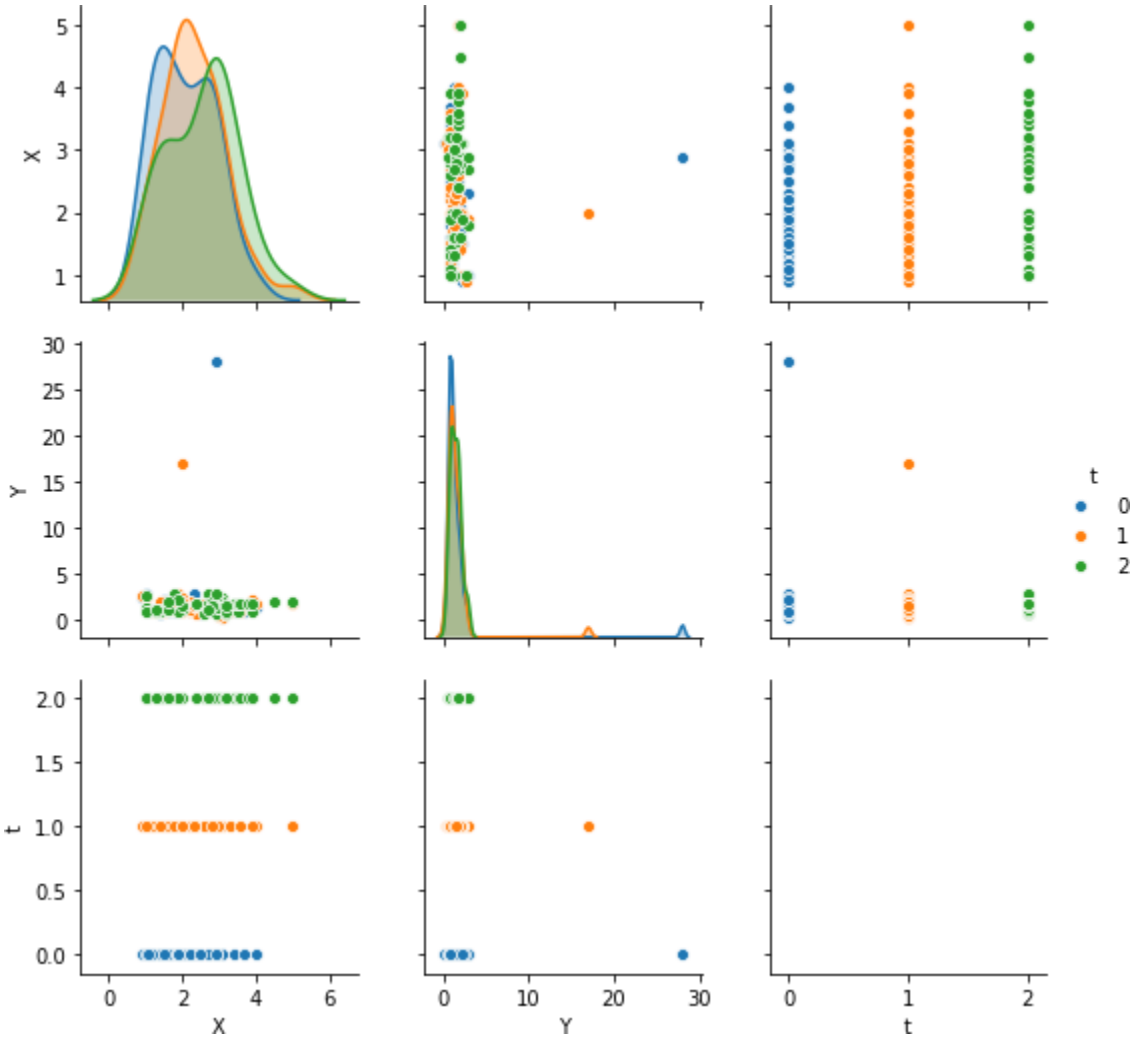
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



[18] :hue='t', diag_kind='hist')

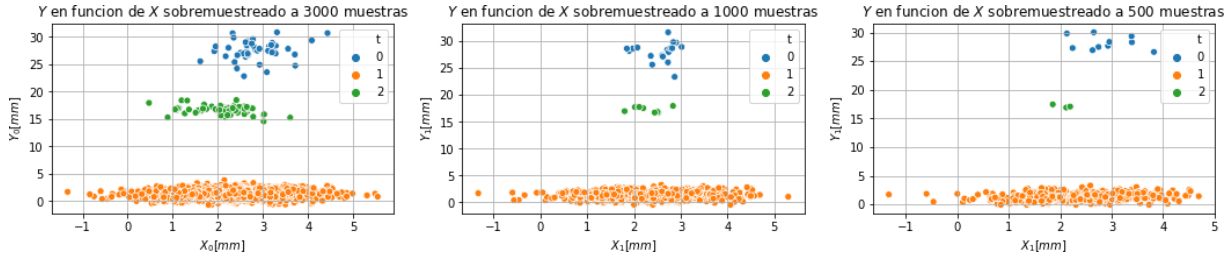


[19]:hue='t')


X
Y
t
0
1
2
t
0
1
2

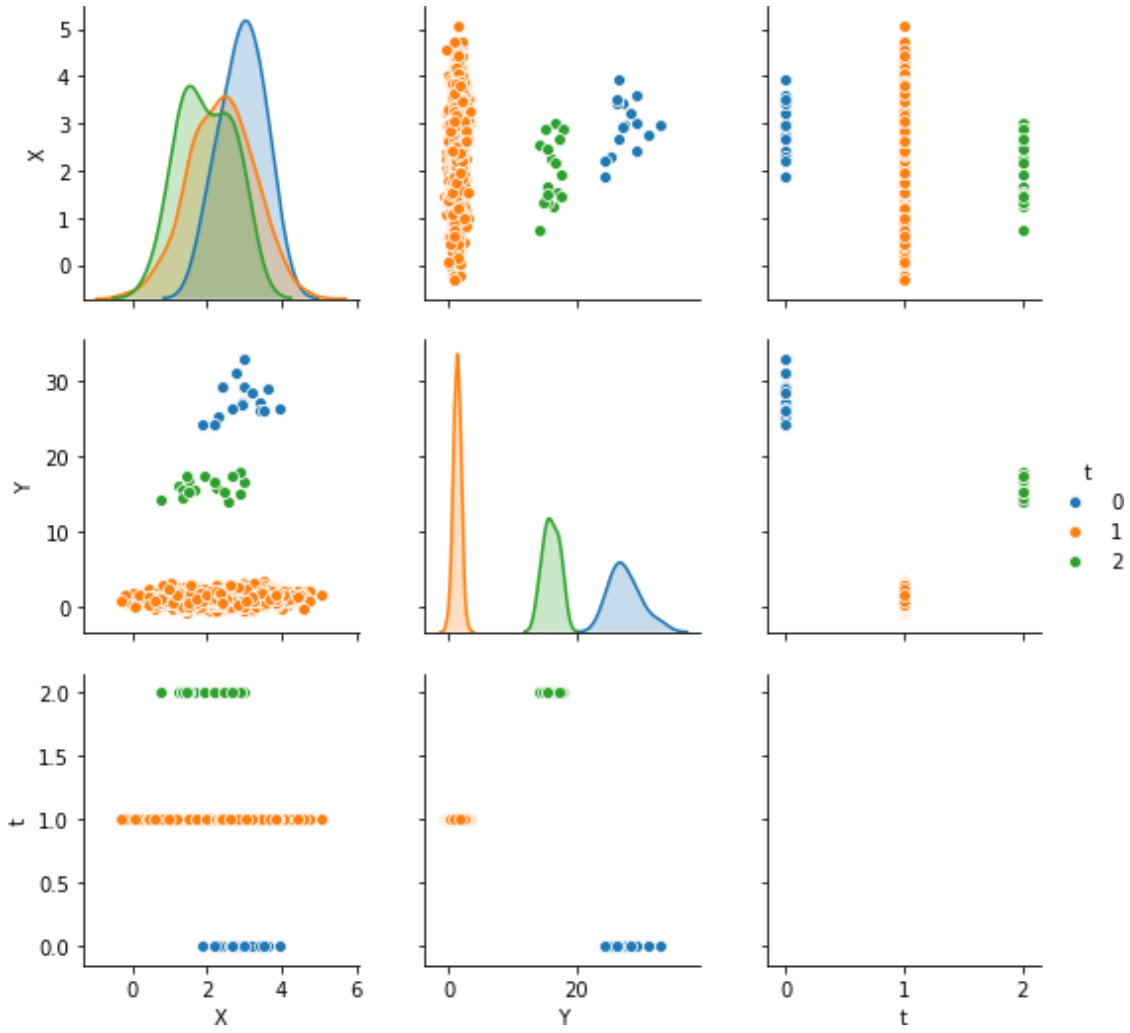
```
[20] : from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
   n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(2),
   init_params="kmeans", max_iter=100, random_state=2).fit(seed2_df[['X', 'Y']])
```

```
[21] : plt.figure(figsize=(18,3)) plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('Jorge_2d_3000_muestras.csv') ax = sns.scatterplot(seed2over_df['X'],
seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
```




```
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```

[22] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']), pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=**True**)
sns.pairplot(seed2over_df, hue='t') plt.show()
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



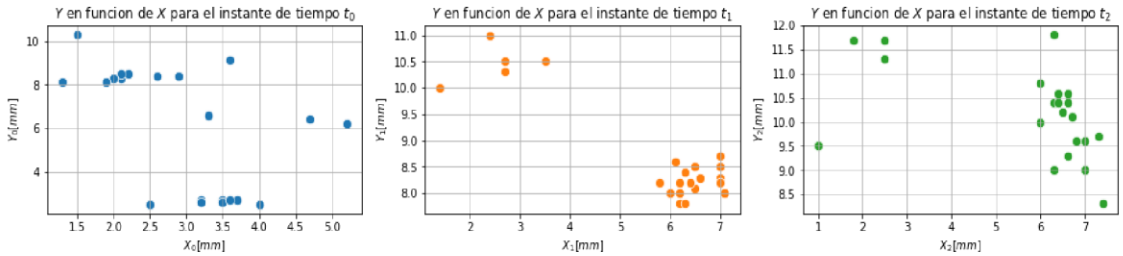
Anexo E Análisis

```
 matplotlib inlin
impo  nu
impo  pand
impo  seabor      s
fr  matplotlib impo  pypl        p
```

```
d2          read_excel r'C:\Users\ufps\Desktop\pasantia\SEMANA3\CRISTINAJORDIN\2D\C
d2  dr     'Z_     'Z_     'Z_      ax       inplac   Tr
d2
```

	X	Y	X	Y	X	Y
	3.2	2.7	7.0	8.3	7.0	9.0
	3.2	2.6	7.0	8.2	6.8	9.6
	3.5	2.7	6.2	8.0	7.0	9.6
	3.5	2.6	7.0	8.5	7.3	9.7
	3.7	2.7	7.1	8.0	6.6	9.3
	3.6	2.7	7.0	8.7	7.4	8.3
	4.0	2.5	6.6	8.3	6.0	10.0
	4.7	6.4	6.5	8.1	6.5	10.2
	5.2	6.2	5.8	8.2	6.3	10.4
	3.3	6.6	6.1	8.6	6.3	10.4
	3.6	9.1	6.3	8.4	6.0	10.8
	2.9	8.4	6.0	8.0	6.4	10.4
	2.6	8.4	6.2	8.2	6.4	10.6
	2.2	8.5	6.4	8.2	6.6	10.4
	2.1	8.3	6.2	7.8	6.6	10.6
	2.2	8.5	6.3	7.8	6.7	10.1
	2.1	8.5	3.5	10.5	6.3	11.8
	2.0	8.3	2.7	10.3	2.5	11.3
	1.3	8.1	2.7	10.5	2.5	11.7
	1.9	8.1	2.4	11.0	1.8	11.7
	1.5	10.3	1.4	10.0	1.0	9.5

[4]:
```
plt.figure(figsize=(18,3))
plt.subplot(131)
ax = sns.scatterplot(d2df['X_0'], d2df['Y_0'],
                     color=sns.color_palette("tab10", n_colors=10)[0],
                     s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

plt.subplot(132)
ax1 = sns.scatterplot(d2df['X_1'], d2df['Y_1'],
                      color=sns.color_palette("tab10", n_colors=10)[1],
                      s=70)
plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_1$')
plt.grid()

plt.subplot(133)
ax2 = sns.scatterplot(d2df['X_2'],d2df['Y_2'],
                      color=sns.color_palette("tab10", n_colors=10)[2],
                      s=70)
plt.xlabel(r'$X_2[mm]$')
plt.ylabel(r'$Y_2[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_2$')
plt.grid()
```



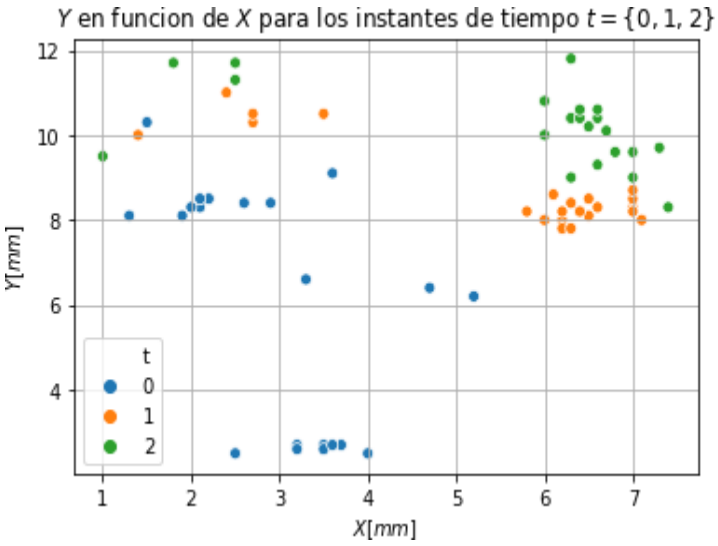
[5]: d2df['t_0'] = np.zeros(len(d2df)) d2df['t_1'] = np.ones(len(d2df)) d2df['t_2'] = np.ones(len(d2df))*2 seed2 = np.vstack((d2df[['X_0', 'Y_0', 't_0']].values, d2df[['X_1', 'Y_1', 't_1']].values, d2df[['X_2', 'Y_2', 't_2']].values)) seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 't']) seed2_df['t'] = seed2_df['t'].astype(int) print('Dimension: ', seed2_df.shape) seed2_df.sample(10)
Dimension: (66, 3)

Out[5]:

	X	Y	t
27	7.1	8.0	1

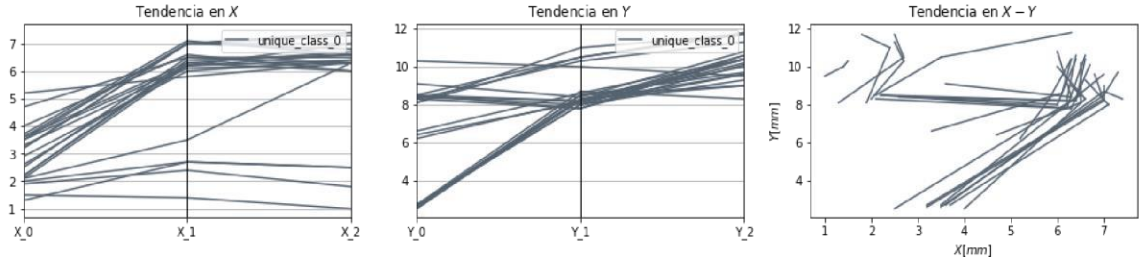
61	6.3	11.8	2
24	7.0	8.2	1
39	3.5	10.5	1
29	6.6	8.3	1
13	2.6	8.4	0
54	6.3	10.4	2
8	4.7	6.4	0
37	6.2	7.8	1
22	6.5	8.5	1

[6]:
```
s    scatterplot    seed2_d '          seed2_d '          h    seed2_d  't
                    palette   s    color_palette    "tab10    n_colors
s    set_palette    "tab10
p    xlabe    r'$X[mm]$'
p    ylabe    r'$Y[mm]$'
p    title    r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$'
p    gri
```



[7]:
```
delta10    seed2_d l    seed2_d  't          '    value    seed2_d  l    seed2_d  '
delta21    seed2_d l    seed2_d  't          '    value    seed2_d  l    seed2_d  '
```

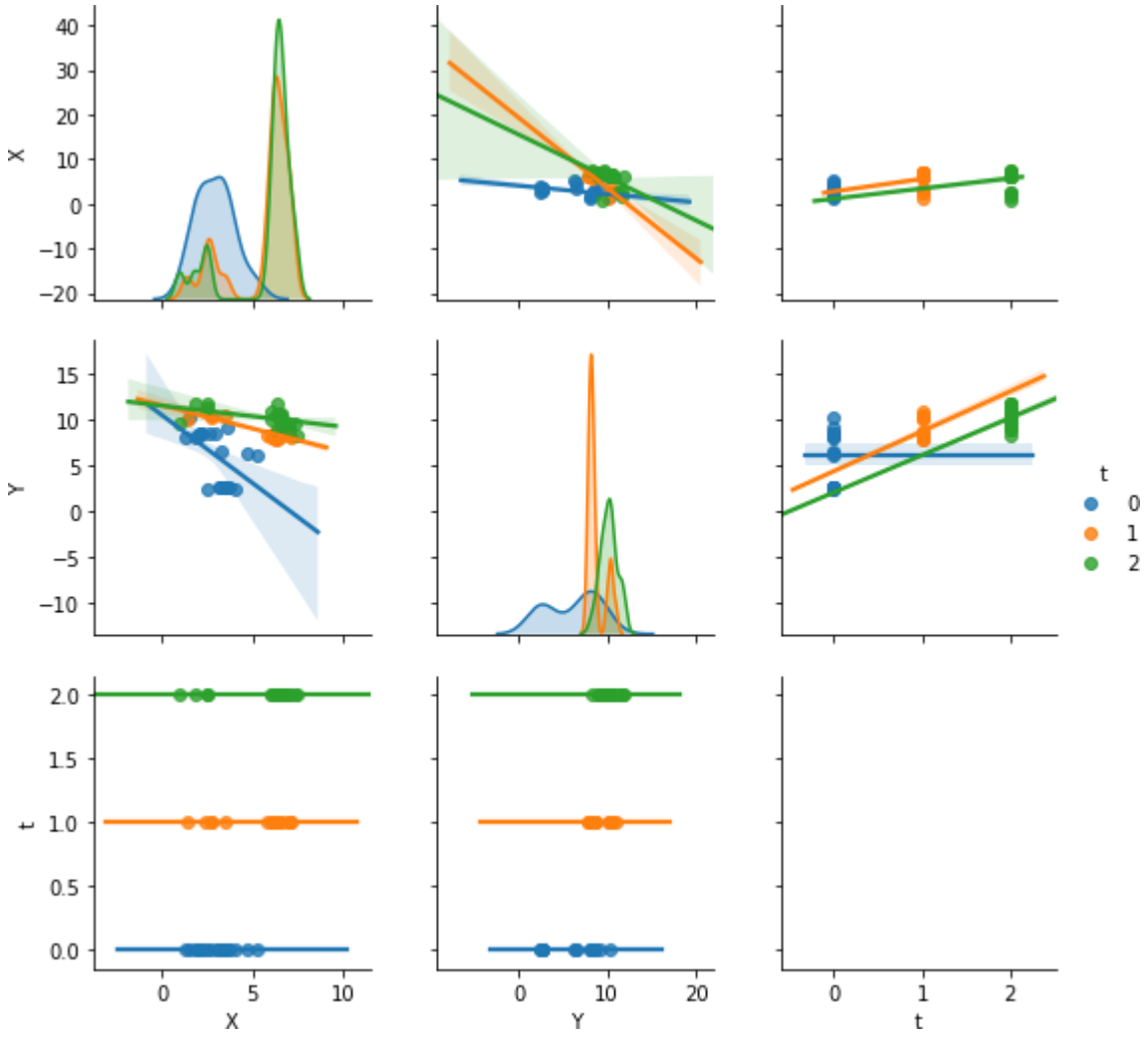
[8]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
for i in range(len(df_0)):
    plt.plot([df_0.loc[i, 'X'], df_1.loc[i, 'X']],
             [df_0.loc[i, 'Y'], df_1.loc[i, 'Y']],
             [df_1.loc[i, 'X'], df_2.loc[i, 'X']],
             [df_1.loc[i, 'Y'], df_2.loc[i, 'Y']], color=('#556270'))
plt.xlabel(r'$X[mm]$')
plt.ylabel(r'$Y[mm]$')
plt.title(r'Tendencia en $X-Y$')
plt.show()
```



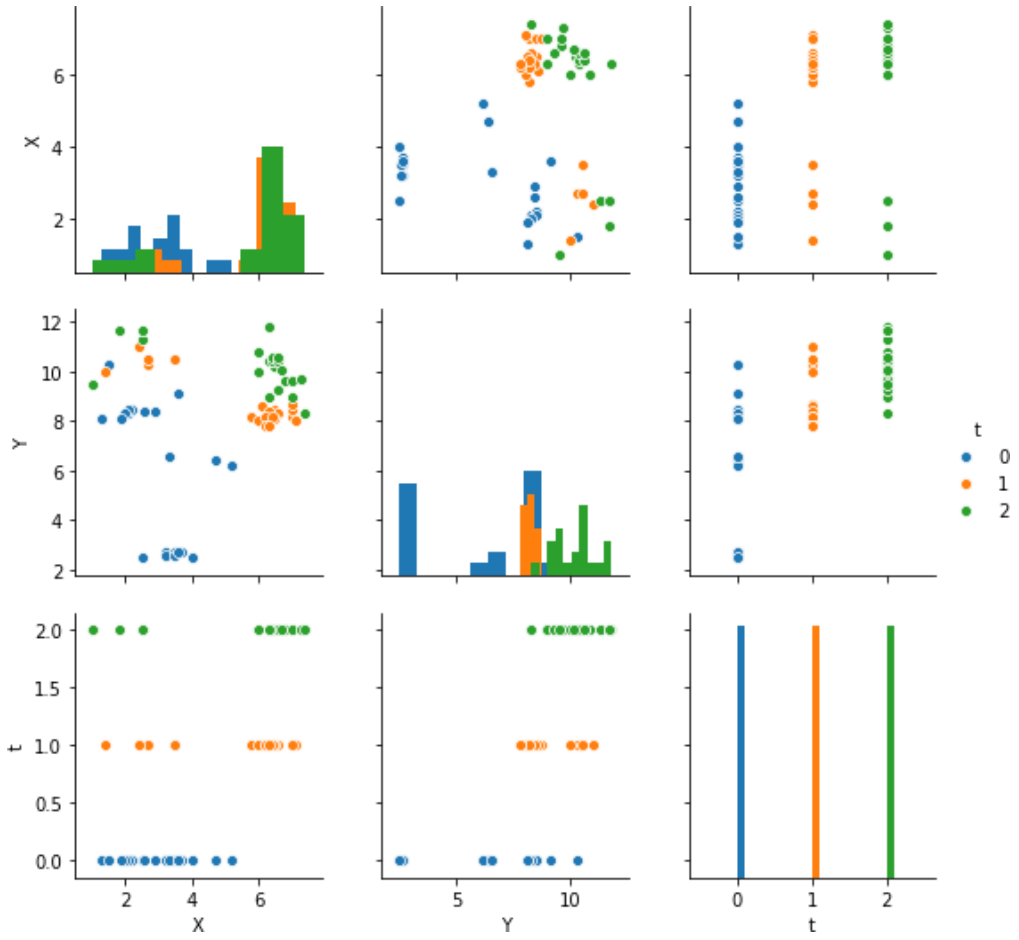
[46] :hue='t', kind="reg")

C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

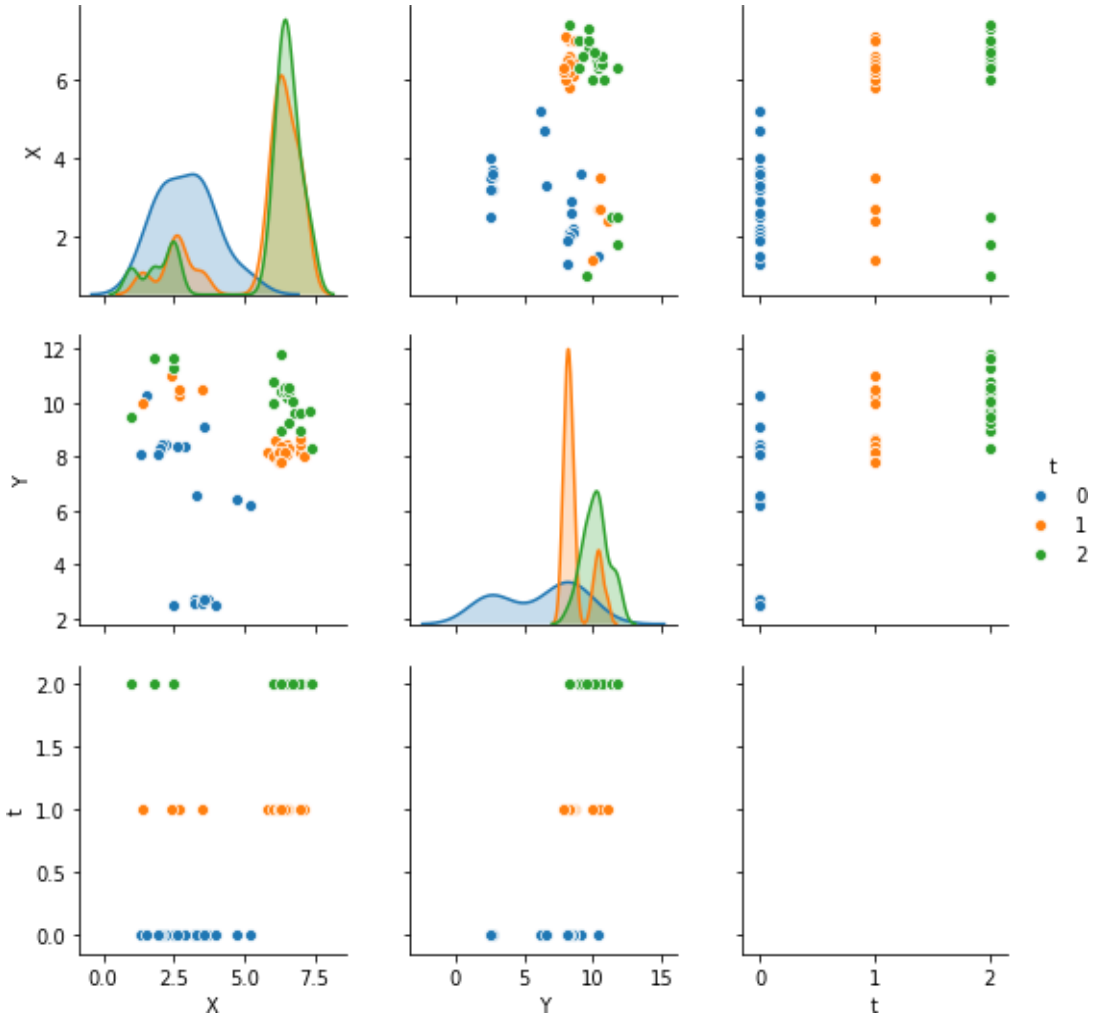
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



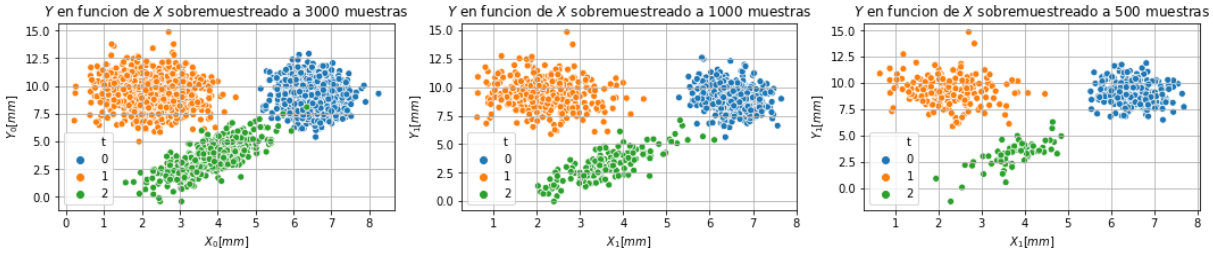
[47] :hue='t', diag_kind='hist')



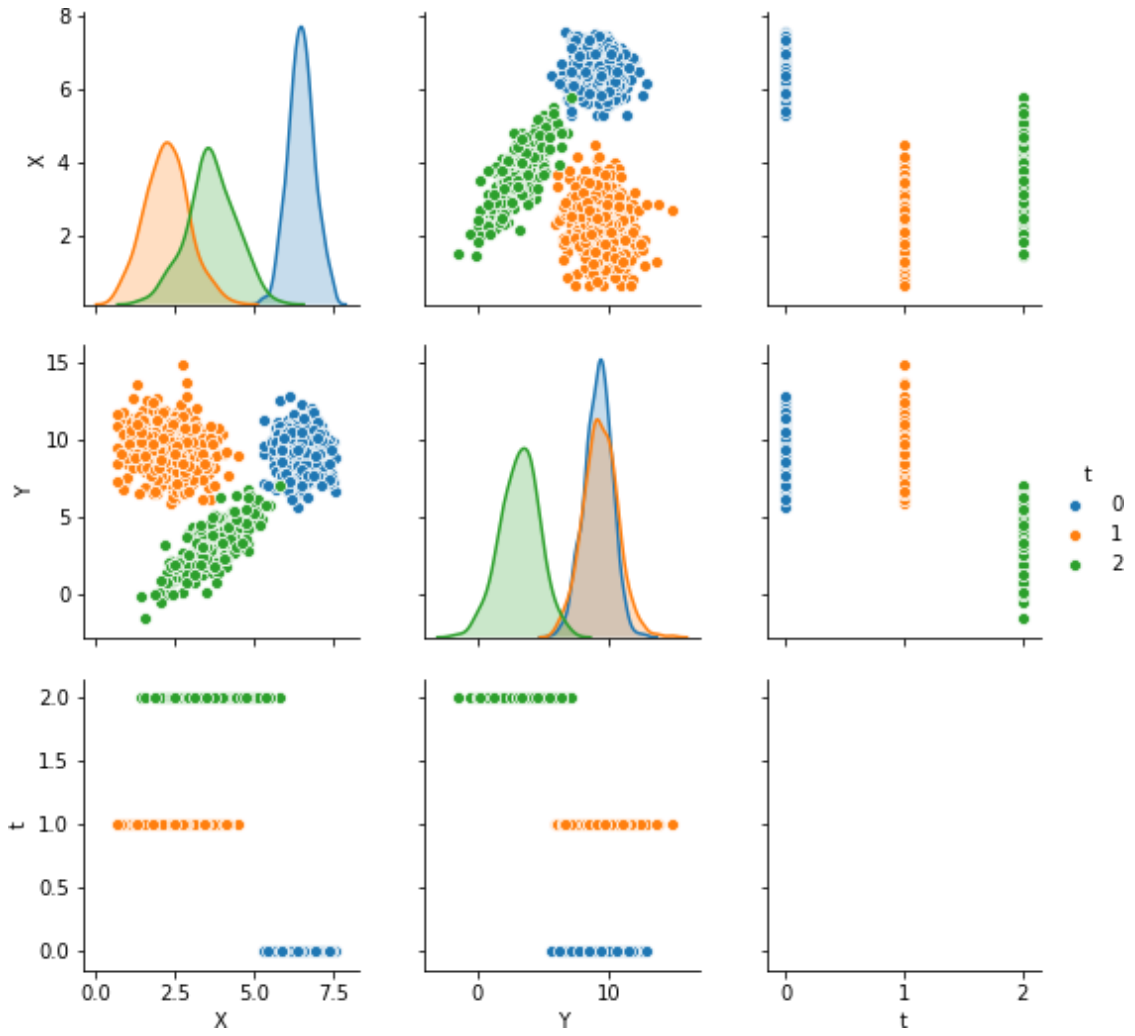
[48]:hue='t')


X
Y
t
0
1
2
0.0
2.5
5.0
7.5
0
5
10
15
0
1
2
0.5
1.5
2
4
6
8
10
12

```
[49] : from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
   n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(2),
   init_params="kmeans", max_iter=100, random_state=2).fit(seed2_df[['X', 'Y']])

[50] : plt.figure(figsize=(18,3)) plt.subplot(131)
X_s, t_s = dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('CRISTINA_2d_3000_muestras.csv') ax = sns.scatterplot(seed2over_df['X'],
seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
plt.subplot(132)
X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
plt.subplot(133)
X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y']),                pd.DataFrame(t_s,
columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True) ax =
sns.scatterplot(seed2over_df['X'], seed2over_df['Y'],                hue=seed2over_df['t'],
            palette=sns.color_palette("tab10", n_colors=3)) plt.xlabel(r'$X_1[mm]$')
plt.ylabel(r'$Y_1[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```



```
[51] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s,
columns=['X', 'Y']),                    pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
sns.pairplot(seed2over_df, hue='t') plt.show()
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:
RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta
* nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning:
invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2
```



Anexo F Análisis

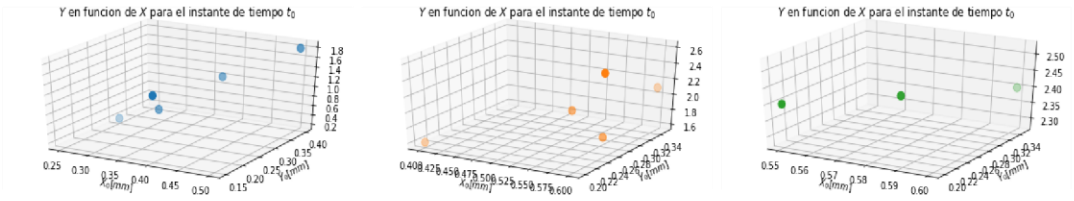
In [3]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

In [4]:
```
from sklearn.base import BaseEstimator, TransformerMixin

class DataFrameImputer(TransformerMixin):

    def __init__(self):
        """Impute missing values.

        Columns of dtype object are imputed with the most frequent value
        in column.

        Columns of other types are imputed with mean of column.

        """

    def fit(self, X, y=None):
        self.fill = pd.Series([X[c].value_counts().index[0]
            if X[c].dtype == np.dtype('O') else X[c].mean() for c in X],
            index=X.columns)

        return self

    def transform(self, X, y=None):

        return X.fillna(self.fill)
```

In [5]:
```
import pandas as pd
d2df = pd.read_excel(r'C:\Users\ufps\Desktop\pasantia\SEMANA3\eduardo\DATOS EXCE
d2df.replace(0, np.nan, inplace=True)
d2df = DataFrameImputer().fit_transform(d2df)
d2df
```

	X_0	Y_0	Z_0	X_1	Y_1	Z_1	X_2	Y_2	Z_2
0	0.35	0.25	0.75	0.40	0.2000	1.600	0.550	0.200	2.4
0.40	0.15	1.45	0.60	0.2500	1.800	0.600	0.350	2.4	
0.25	0.30	0.20	0.60	0.3500	2.100	0.575	0.275	2.4	
0.40	0.35	1.20	0.60	0.2500	2.600	0.575	0.275	2.4	
0.50	0.40	1.80	0.55	0.2625	2.025	0.575	0.275	2.4	

In [6]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
ax.scatter(d2df['X_0'], d2df['Y_0'], d2df['Z_0'],
           color=sns.color_palette("tab10", n_colors=10)[0],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
ax.scatter(d2df['X_1'], d2df['Y_1'], d2df['Z_1'],
           color=sns.color_palette("tab10", n_colors=10)[1],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
ax.scatter(d2df['X_2'], d2df['Y_2'], d2df['Z_2'],
           color=sns.color_palette("tab10", n_colors=10)[2],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()
plt.tight_layout()
plt.show()
```



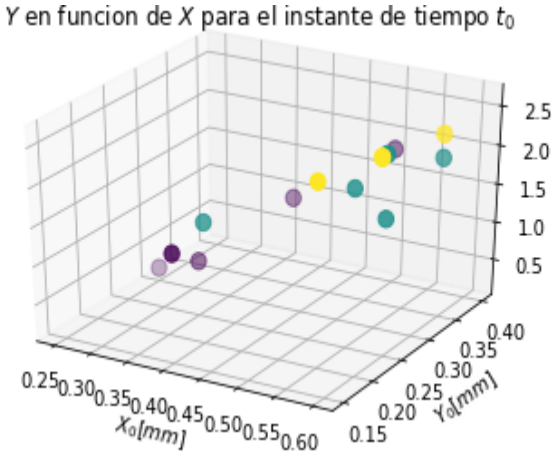
In [8]: d2df['t_0'] = np.zeros(len(d2df)) d2df['t_1'] = np.ones(len(d2df)) d2df['t_2'] = np.ones(len(d2df))*2
seed2 = np.vstack((d2df[['X_0', 'Y_0', 'Z_0', 't_0']].values, d2df[['X_1', 'Y_1', 'Z_1', 't_1']].values, d2df[['X_2', 'Y_2', 'Z_2', 't_2']].values)) seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 'Z', 't']) seed2_df['t'] = seed2_df['t'].astype(int) print('Dimension: ', seed2_df.shape)
seed2_df.sample(5)
Dimension: (15, 4)

Out[8]:

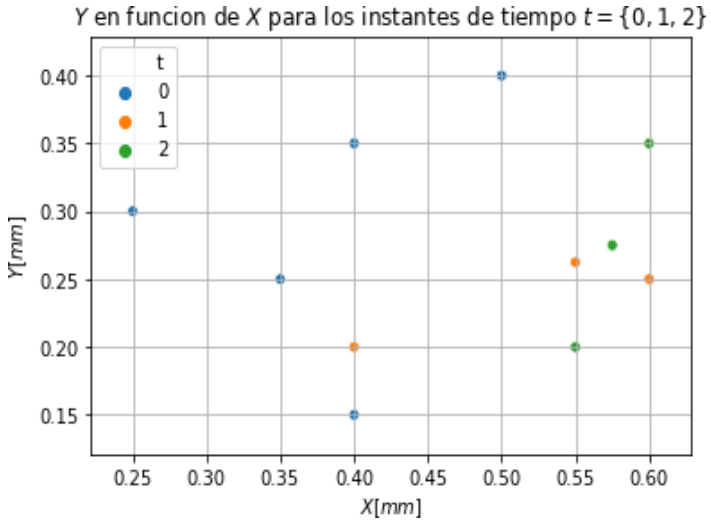
	X	Y	Z	t
12	0.575	0.275	2.40	2
7	0.600	0.350	2.10	1

11	0.600	0.350	2.40	2
5	0.400	0.200	1.60	1
0	0.350	0.250	0.75	0

[9]:
```
f       p    figur
        f    add_subplot 1     projection  '3
     scatter   seed2_d  '        seed2_d  '        seed2_d '
                    seed2_d 't
p    xlabe   r'$X_0[mm]$'
p    ylabe   r'$Y_0[mm]$'
p    title   r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$'
p    gri
```

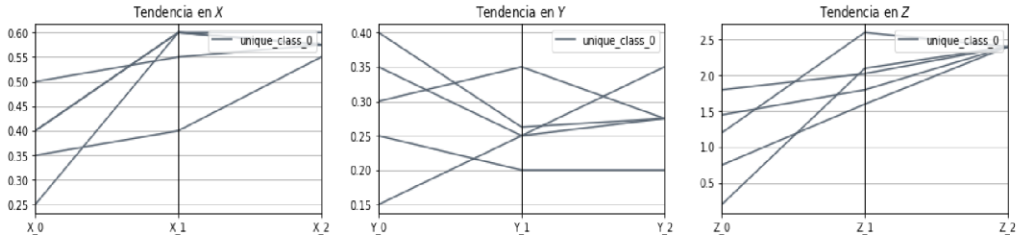


```
[10]:          s    scatterplot    seed2_d '          seed2_d '          h    seed2_d 't
                                   palette   s     color_palette    "tab10     n_color
    s     set_palette    "tab10
    p     xlabe r'$X[mm]$
    p     ylabe    r'$Y[mm]$
    p     title    r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$'
    p     gr
```




```
In [11]: delta10 = seed2_df.loc[seed2_df['t'] == 1]['Y'].values-seed2_df.loc[seed2_df['t' delta21 =
seed2_df.loc[seed2_df['t'] == 2]['Y'].values-seed2_df.loc[seed2_df['t'
```

In [12]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
pd.plotting.parallel_coordinates(
        df[['Z_0','Z_1','Z_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Z$')
plt.show()
```

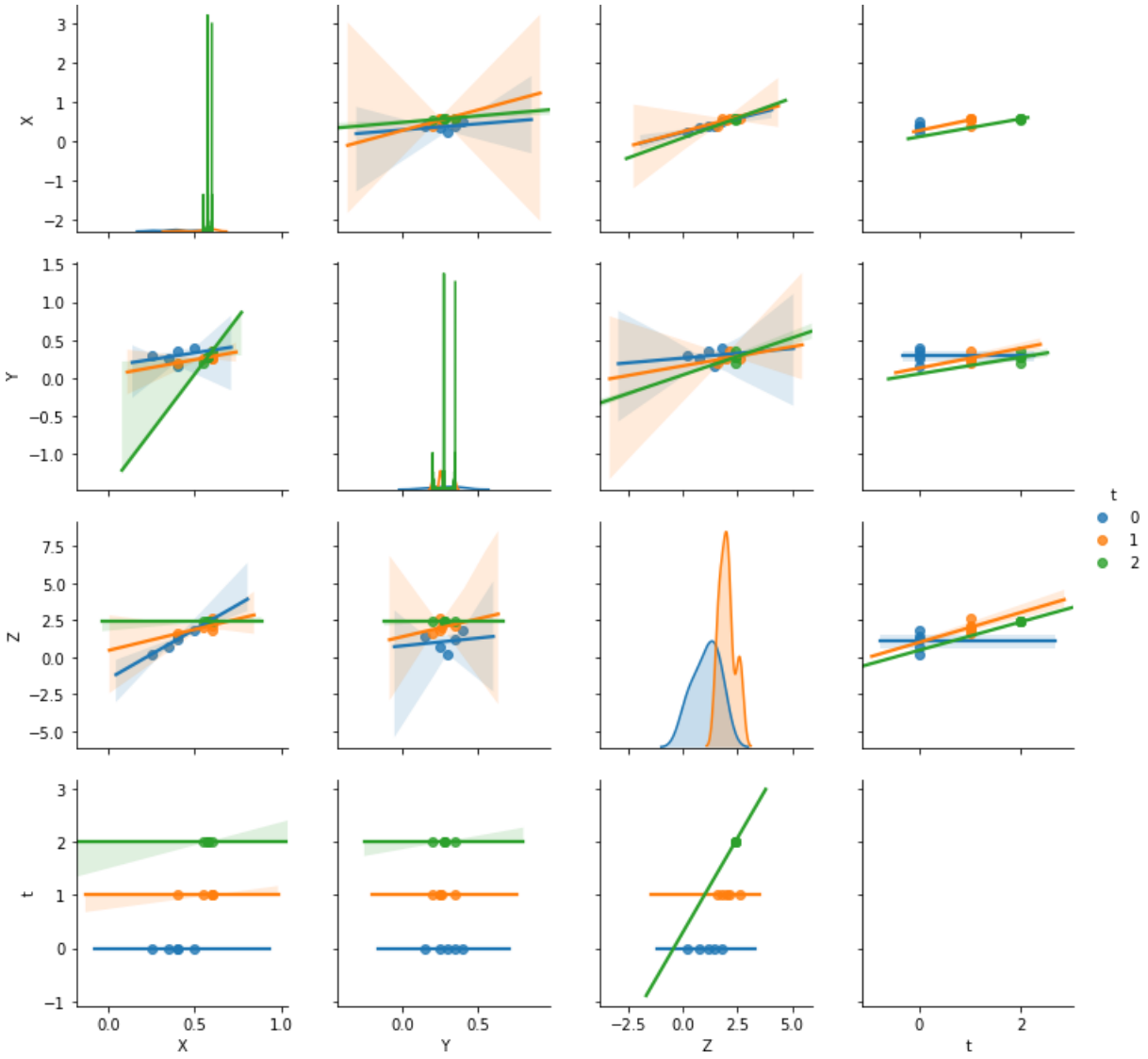


C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:

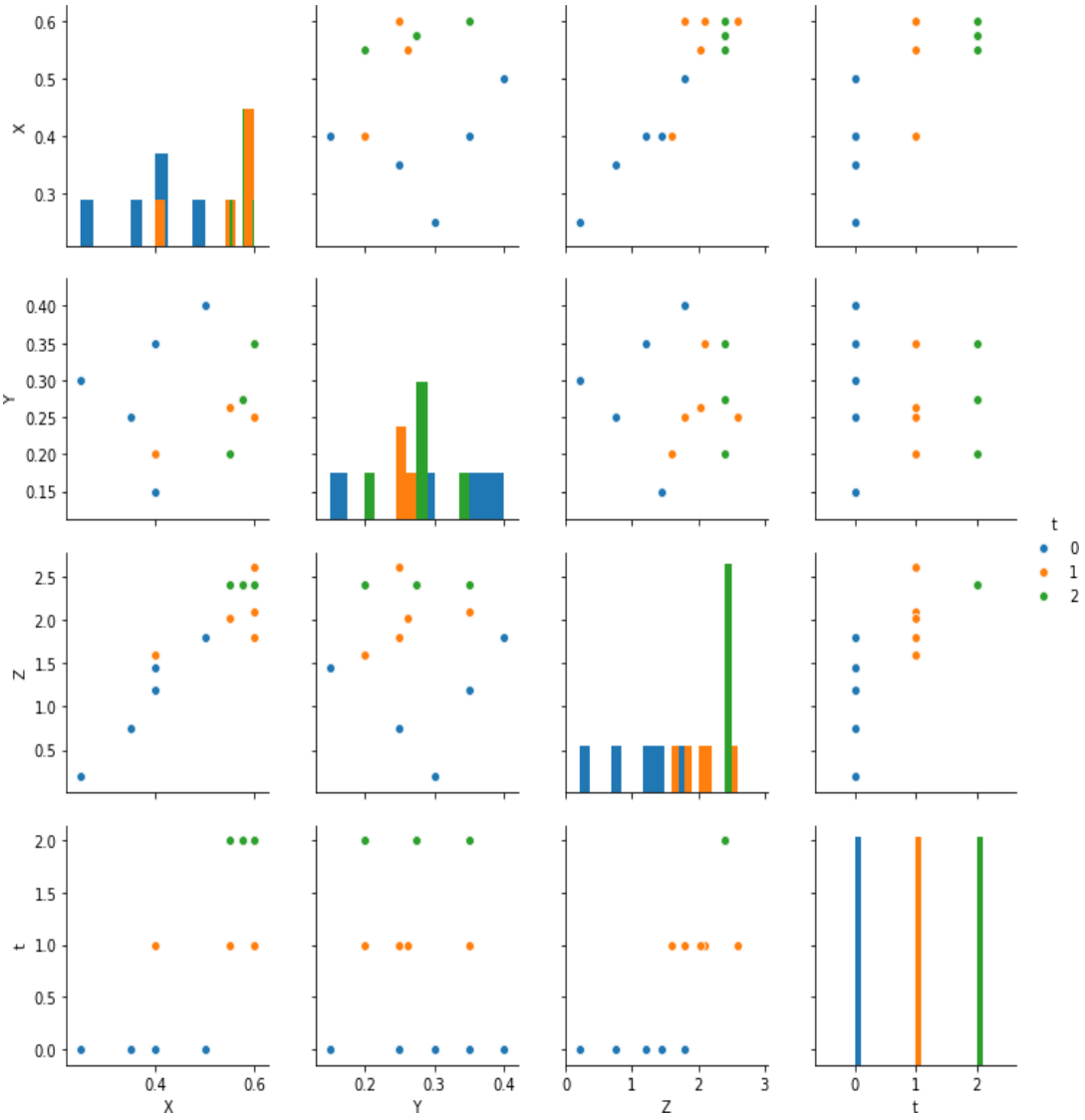
[13]:
```
s    pairplot    seed2_df    h    't'    kin    "reg
p    sh
```

RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

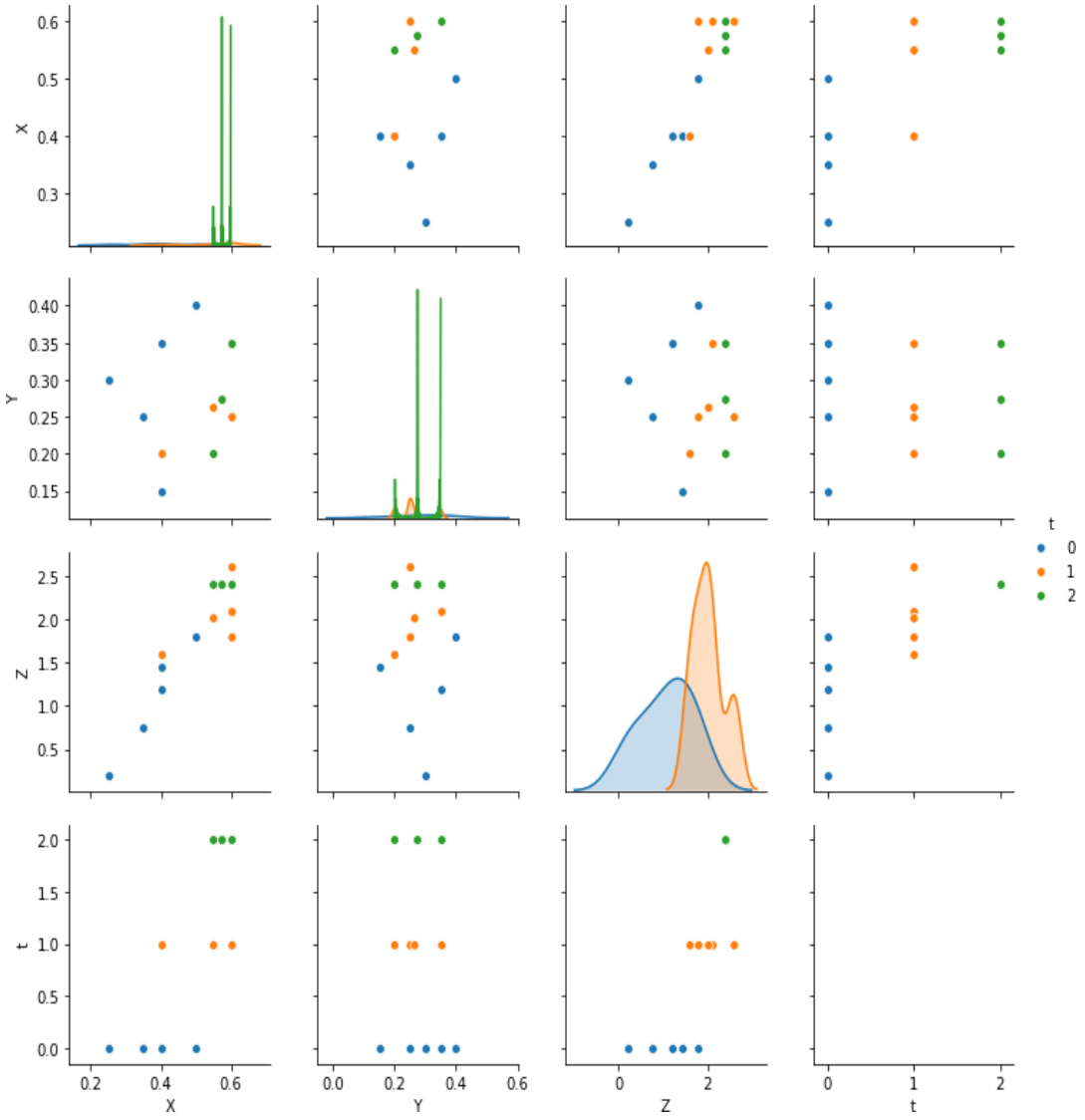
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2


X
Y
Z
t
t
0
1
2

[14]:
```
s    pairplot seed2_d    h    't    diag_kind 'hist'
p    sh
```

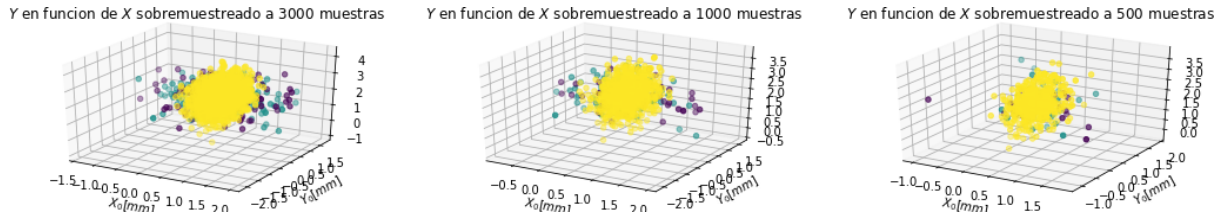


[15]:
```
s    pairplot seed2_d    h    't
p    sh
```

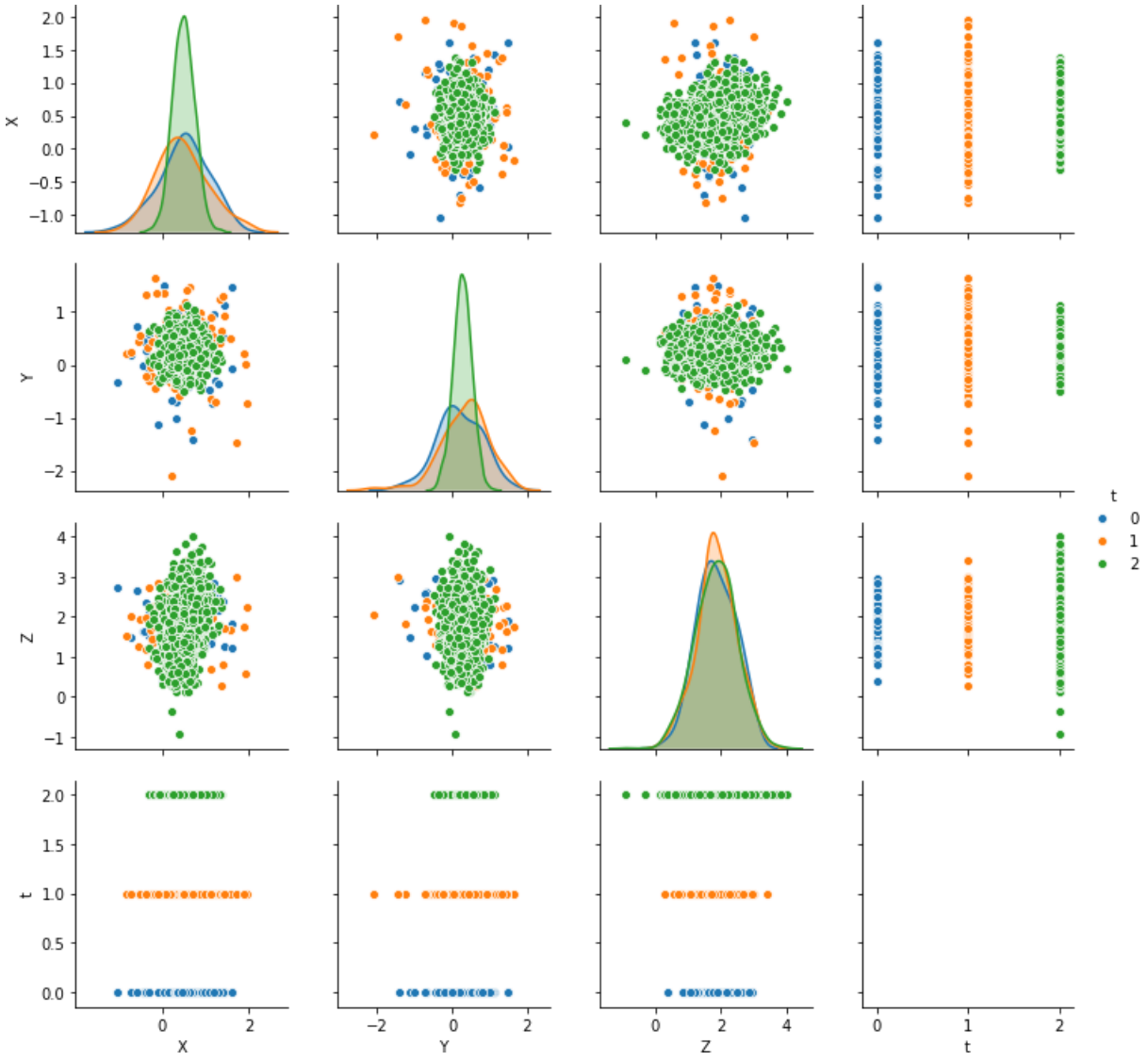


```
[16]: from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
   n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(3),     init_params="kmeans", max_iter=100,
random_state=2).fit(seed2_df[['X', 'Y',
In [17]: X_s, t_s = dpgmm.sample(n_samples=3000) X_s
Out[17]: array([[ 0.98764973,  0.14263627,  2.90209521],       [ 1.11247567, 0.90169151,  2.46796706],
       [-0.82174025, 0.20479016,  1.52372001],       ...,
       [ 0.59882722, 0.61996432,  1.58400534],
       [ 0.67901089, 0.36710454,  2.86298968],       [ 0.47618902, 0.1151856 , 1.57869353]])
```

```
[18] :     fig = plt.figure(figsize=(18,3)) ax = fig.add_subplot(131, projection='3d') X_s, t_s =
dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
ax = fig.add_subplot(132, projection='3d') X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
ax = fig.add_subplot(133, projection='3d') X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],          c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```



```
[19] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s,
columns=['X', 'Y', 'Z']),                    pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
sns.pairplot(seed2over_df, hue='t') plt.show()
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:
RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta
* nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning:
invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2
```



Anexo G Análisis

```
 matplotlib inlin
impo  nu
impo  pand
impo  seabor      s
f     matplotlib impo  pypl        p
```

```
f     sklear   ba    impo  BaseEstimator TransformerMixin

cla   DataFrameImputerTransformerMixin

    d    _init___se
         """Impute missing values.

         Columns of dtype object are imputed with the most frequent value
         in column.

         Columns of other types are imputed with mean of column.

         """

         d   f    se          N
             se    fil         Serie          value_counts(   ind
                           dty          dty   '     e          m      f
                 ind       colum

             retu  se

         d   transform  se          N

             retu     fillna   se    fil
```

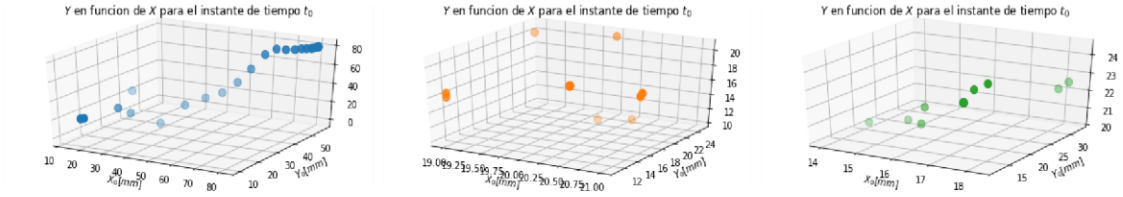
19	10.5	22.50	19.000	11.0000	16.00000	15.0000	22.00	20.50000
13	41.0	23.40	21.000	16.0000	17.20000	16.0000	17.00	21.14000
20	26.0	20.00	21.000	16.3000	17.50000	16.0000	17.80	22.00000
21	31.0	10.30	20.000	22.0000	10.17000	18.0000	12.00	24.15000
30	36.0	-2.40	20.400	22.6000	10.50000	18.3000	12.40	24.50000
36	42.0	14.80	19.000	24.0000	20.14000	18.0000	32.00	22.12000
41	47.0	19.90	20.000	24.8000	20.60000	18.2000	32.60	22.50000
46	50.0	24.90	19.925	18.4625	16.07625	16.6875	20.85	22.13625
51	52.0	36.10	19.925	18.4625	16.07625	16.6875	20.85	22.13625
56	53.0	51.60	19.925	18.4625	16.07625	16.6875	20.85	22.13625

[4]:
```
impo   panda
d2          read_excel   'jhon3d.xlsx'
d2    replace          n      inplace   Tr
d2       DataFrameImputer () fit_transform      d2
d2
```

Out[4]:

	X	Y	Z	X	Y	Z	X	Y	Z
		1	22.	19.0	11.000	16.5000	14.000	21.	20.1800
61	54.0	67.70	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
65	55.0	73.90	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
69	55.0	74.40	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
72	56.0	74.70	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
74	57.0	75.40	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
76	57.0	76.20	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
78	57.0	77.10	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
79	57.0	77.80	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
80	57.0	78.60	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
81	57.0	79.30	19.925	18.4625	16.07625	16.6875	20.85	22.13625	
81	57.0	80.00	19.925	18.4625	16.07625	16.6875	20.85	22.13625	

[5]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
ax.scatter(d2df['X_0'], d2df['Y_0'], d2df['Z_0'],
           color=sns.color_palette("tab10", n_colors=10)[0],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
ax.scatter(d2df['X_1'], d2df['Y_1'], d2df['Z_1'],
           color=sns.color_palette("tab10", n_colors=10)[1],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
ax.scatter(d2df['X_2'], d2df['Y_2'], d2df['Z_2'],
           color=sns.color_palette("tab10", n_colors=10)[2],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()
plt.tight_layout()
plt.show()
```



[7]:
```
d2df['t_0'] = np.zeros(len(d2df))
d2df['t_1'] = np.ones(len(d2df))
d2df['t_2'] = np.ones(len(d2df))*2
seed2 = np.vstack((d2df[['X_0', 'Y_0', 'Z_0', 't_0']].values,
                   d2df[['X_1', 'Y_1', 'Z_1', 't_1']].values,
                   d2df[['X_2', 'Y_2', 'Z_2', 't_2']].values))
seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 'Z', 't'])
seed2_df['t'] = seed2_df['t'].astype(int)
print('Dimension: ', seed2_df.shape)
seed2_df.sample(30)
```

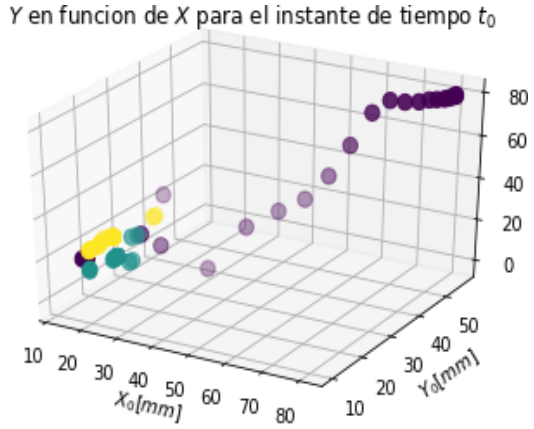
```
Dimension:  (66, 4)
```

Out[7]:

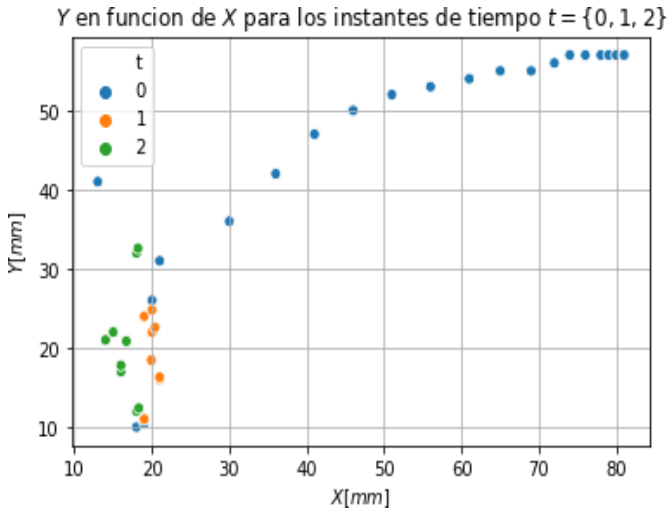
	X	Y	Z	t
46	16.0000	17.0000	21.14000	2
31	19.9250	18.4625	16.07625	1
26	20.0000	22.0000	10.17000	1
3	20.0000	26.0000	20.00000	0
57	16.6875	20.8500	22.13625	2
41	19.9250	18.4625	16.07625	1
48	18.0000	12.0000	24.15000	2
8	46.0000	50.0000	24.90000	0
39	19.9250	18.4625	16.07625	1
36	19.9250	18.4625	16.07625	1
35	19.9250	18.4625	16.07625	1
62	16.6875	20.8500	22.13625	2
60	16.6875	20.8500	22.13625	2
15	74.0000	57.0000	75.40000	0
4	21.0000	31.0000	10.30000	0
25	21.0000	16.3000	17.50000	1
14	72.0000	56.0000	74.70000	0
55	16.6875	20.8500	22.13625	2
58	16.6875	20.8500	22.13625	2
38	19.9250	18.4625	16.07625	1
16	76.0000	57.0000	76.20000	0
33	19.9250	18.4625	16.07625	1
43	19.9250	18.4625	16.07625	1
21	81.0000	57.0000	80.00000	0
12	65.0000	55.0000	73.90000	0
50	18.0000	32.0000	22.12000	2
29	20.0000	24.8000	20.60000	1

78.00	57.00	77.100
19.92	18.46	16.076
20.40	22.60	10.500

```
f       p    figur
       f    add_subplot 1      projection '3
    scatter  seed2_   '        seed2_   '        seed2_   '
                seed2_   't
p   xlab    r'$X_0[mm]
p   ylab    r'$Y_0[mm]
p   title   r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$'
p   gr
```

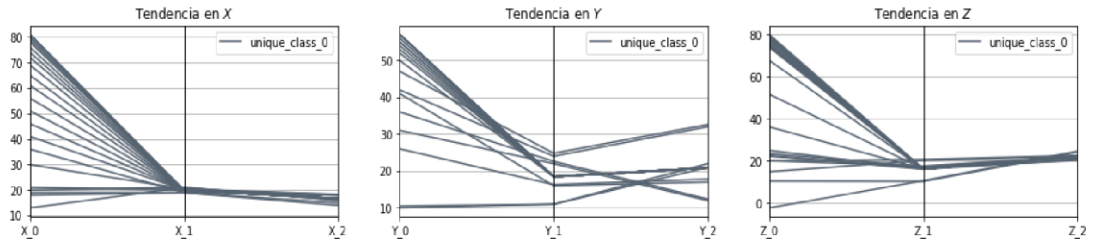



```
        s    scatterplot  seed2_   '       seed2_   '       h    seed2_   't
                          palette  s    color_palette  "tab10    n_color
s   set_palette "tab10
p   xlab    r'$X[mm]
p   ylab    r'$Y[mm]
p   title   r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$'
p   gr
```



In [10]:
```
delta10 = seed2_df.loc[seed2_df['t'] == 1]['Y'].values-seed2_df.loc[seed2_df['t'
delta21 = seed2_df.loc[seed2_df['t'] == 2]['Y'].values-seed2_df.loc[seed2_df['t'
```

In [11]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
pd.plotting.parallel_coordinates(
        df[['Z_0','Z_1','Z_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Z$')
plt.show()
```

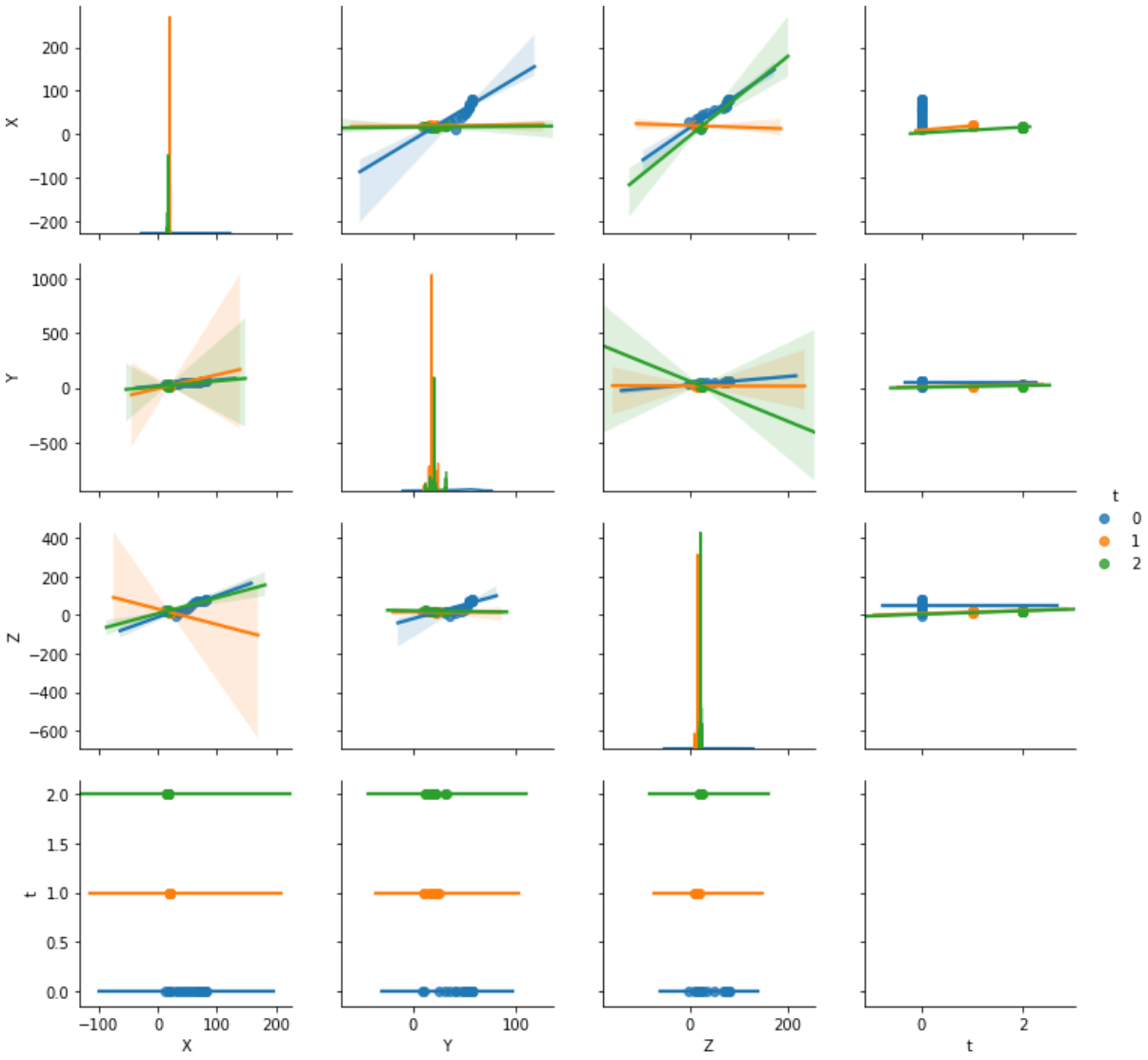


C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:

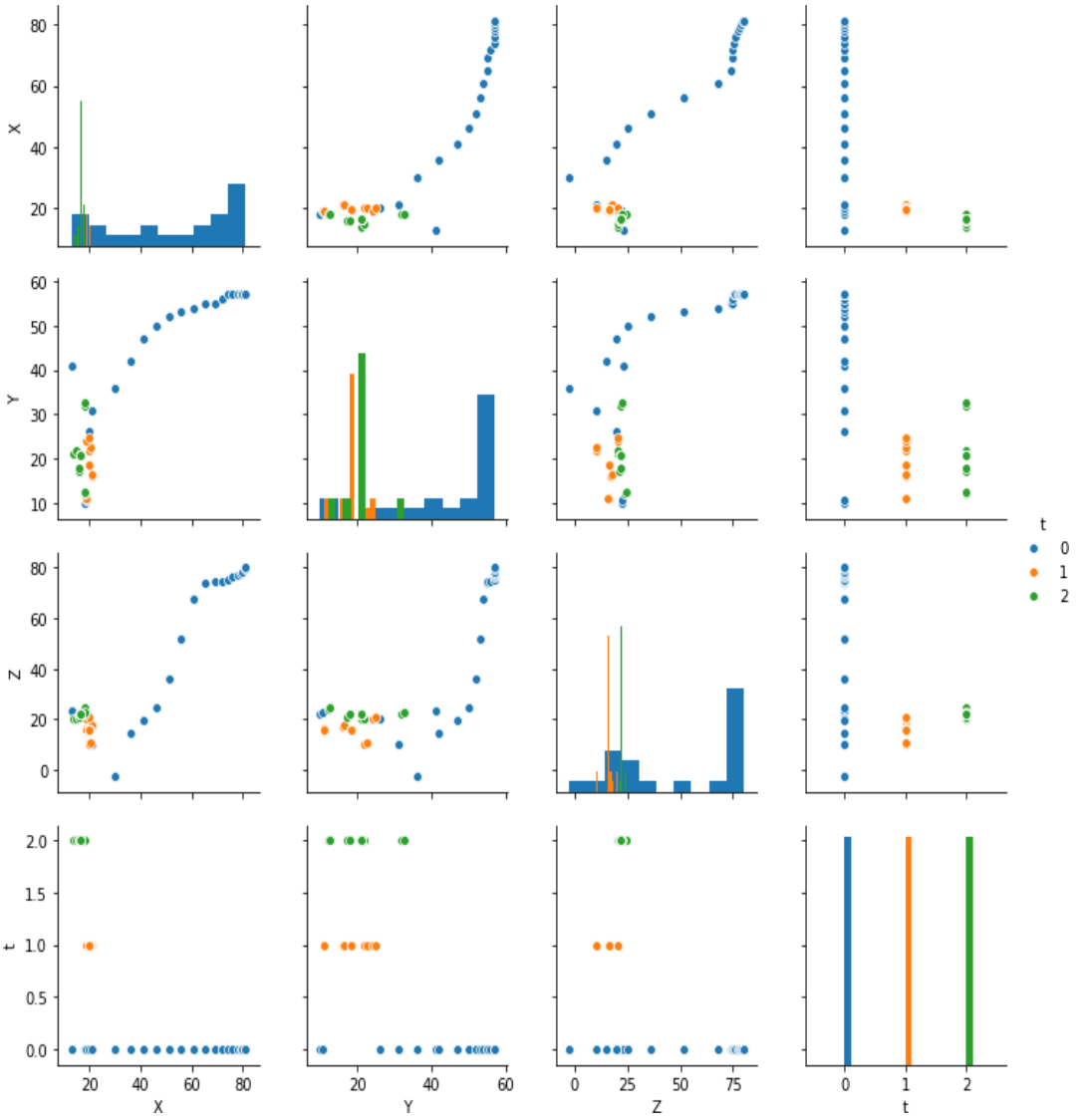
[12]:
```
s    pairplot   seed2_d    h    't     ki     "reg
p    sh
```

RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

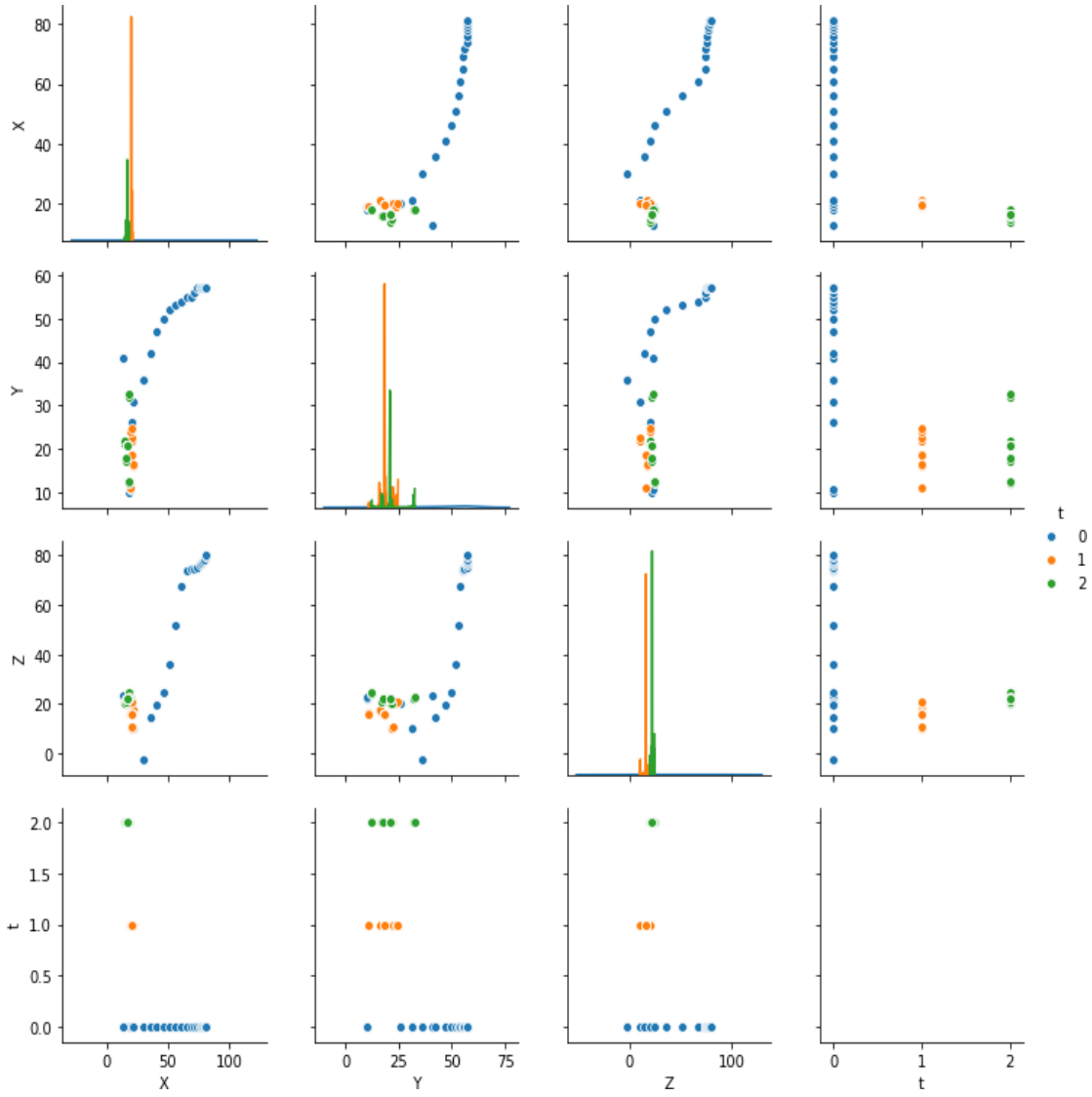
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2


X
Y
Z
t
t
0
1
2

[13]:
```
s    pairplot seed2_d    h   't    diag_kin  'hist'
p    sh
```

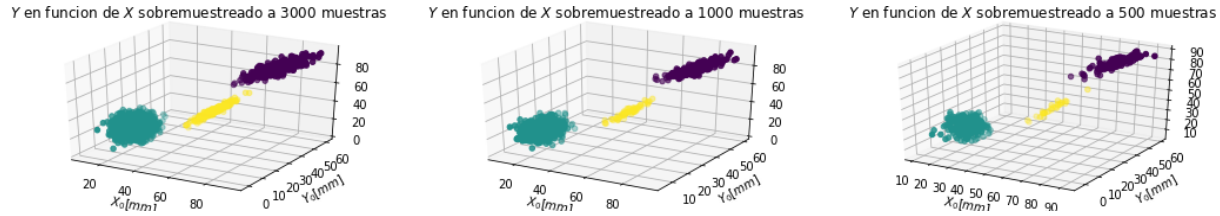


[14]:
```
s    pairplot    seed2_d    h    't
p    sh
```



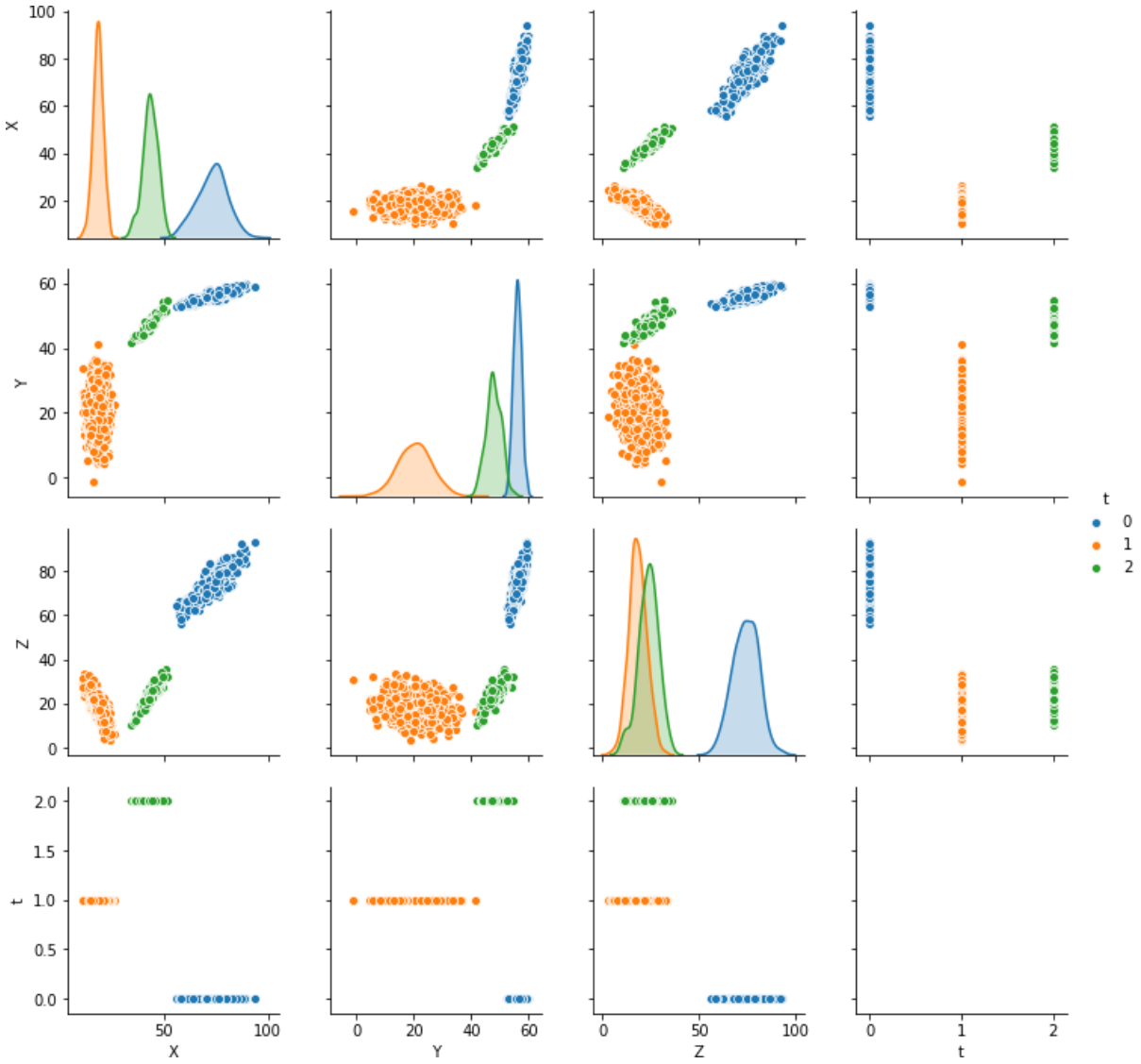
```
[15]: from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
   n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(3),     init_params="kmeans", max_iter=100,
random_state=2).fit(seed2_df[['X', 'Y',
In [16]: X_s, t_s = dpgmm.sample(n_samples=3000) X_s
Out[16]: array([[18.96393831, 31.82302871, 17.88556284],       [16.5562739 , 29.22372678,
19.30541635],
       [22.69617799, 12.71802493, 11.30405792],       ...,
       [41.15536813, 45.17974038, 23.16450385],
       [49.93331263, 51.65176915, 33.56857918],       [42.0565722 , 47.32189933, 21.15138055]])
```

```
[17] :      fig = plt.figure(figsize=(18,3)) ax = fig.add_subplot(131, projection='3d') X_s, t_s =
dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
ax = fig.add_subplot(132, projection='3d') X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
ax = fig.add_subplot(133, projection='3d') X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],          c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
```




```
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```

[18] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']), pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=**True**)
sns.pairplot(seed2over_df, hue='t') plt.show()
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



```
 matplotlib inlin
impo  nu
impo  pand
impo  seabor      s
fr   matplotlib impo  pypl        p
```

```
fr   sklear   ba   impo  BaseEstimator  TransformerMixin

cla   DataFrameImputerTransformerMixin

    d   _init___se
        """Impute missing values.

        Columns of dtype object are imputed with the most frequent value
        in column.

        Columns of other types are imputed with mean of column.

        """

        d   f    se          No
            se    fil          Serie          value_counts(   ind
                        dty          dty    '     el        m      f
                ind      colum

            retu   se

        d   transform  se          No

            retu     fillna   se    fil
```

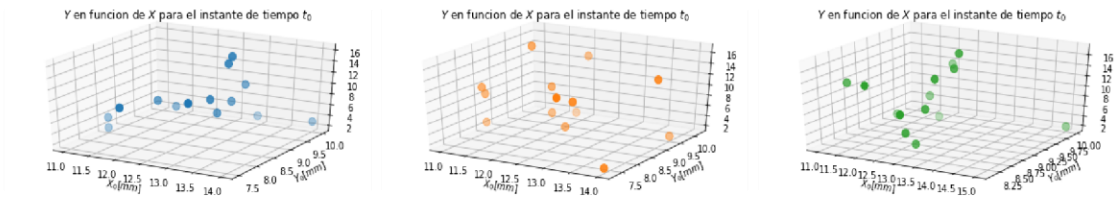
```
impo   pand
d2          read_excel 'jorge3d.xlsx'
d2   replac          n      inplac   Tr
d2      DataFrameImpute(r fit_transform d2
d2
```

Out[3]

	X	Y	Z	X	Y	Z	X	Y	Z
				1			1		
11	8.5	2.0	11.5	9.2	16.5	13.0	9.3	17.0
13	9.0	15.0	14.0	9.5	2.0	12.0	9.7	2.0
13	9.5	10.0	14.0	7.3	2.0	15.0	10.0	2.0
12	7.5	10.0	12.5	8.6	4.0	13.0	8.2	6.0
11	8.5	4.0	12.0	9.0	5.0	12.0	9.5	7.0
13	9.1	8.0	12.0	9.8	3.0	11.5	9.2	4.0

13	9.9	3.0	11.0	8.5	8.5	13.0	8.6	9.0
14	10.0	3.0	13.0	7.5	13.0	13.0	9.2	14.5
13	8.7	7.0	12.0	9.0	10.0	12.0	10.0	11.5
13	7.9	11.0	11.0	8.4	10.0	13.0	8.1	10.0
12	9.0	6.0	14.0	9.0	14.0	13.0	8.8	14.0
13	8.5	10.0	12.0	10.2	13.0	11.0	8.6	12.0
13	9.1	16.0	13.0	8.0	11.0	12.0	8.2	14.0

[4]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
ax.scatter(d2df['X_0'], d2df['Y_0'], d2df['Z_0'],
           color=sns.color_palette("tab10", n_colors=10)[0],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
ax.scatter(d2df['X_1'], d2df['Y_1'], d2df['Z_1'],
           color=sns.color_palette("tab10", n_colors=10)[1],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
ax.scatter(d2df['X_2'], d2df['Y_2'], d2df['Z_2'],
           color=sns.color_palette("tab10", n_colors=10)[2],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()
plt.tight_layout()
plt.show()
```



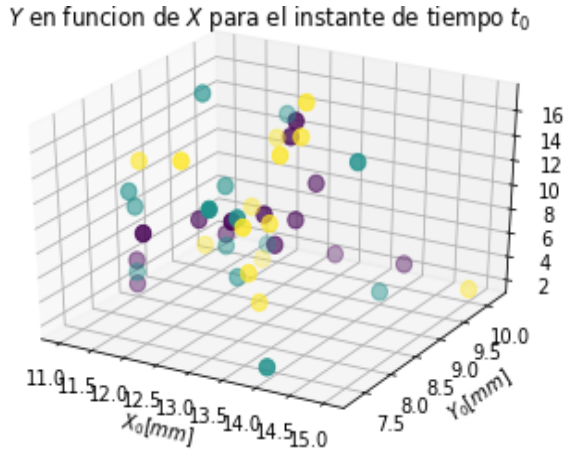
[5]: d2df['t_0'] = np.zeros(len(d2df)) d2df['t_1'] = np.ones(len(d2df)) d2df['t_2'] = np.ones(len(d2df))*2 seed2 = np.vstack((d2df[['X_0', 'Y_0', 'Z_0', 't_0']].values, d2df[['X_1', 'Y_1', 'Z_1', 't_1']].values, d2df[['X_2', 'Y_2', 'Z_2', 't_2']].values)) seed2_df = pd.DataFrame(seed2,

```
columns=['X', 'Y', 'Z', 't']) seed2_df['t'] = seed2_df['t'].astype(int) print('Dimension: ', seed2_df.shape)
seed2_df.sample(20)
Dimension:  (42, 4)
```

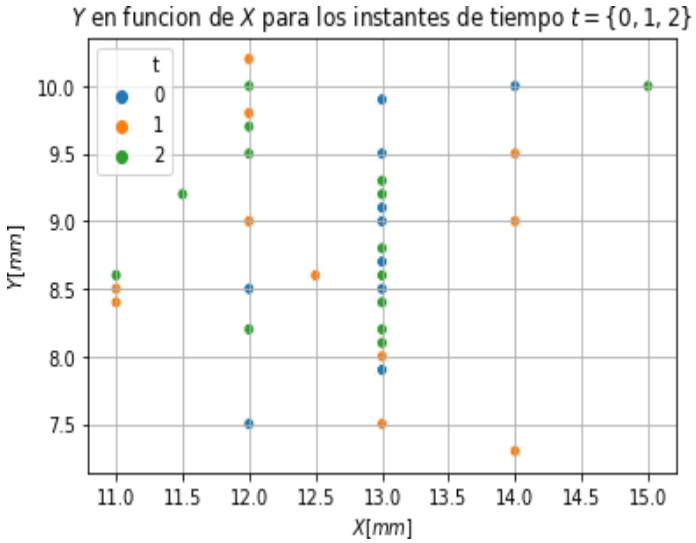
Out[5]:

	X	Y	Z	t	
32	13.0	8.2	6.0	2
4	12.0	7.5	10.0	0
36	13.0	9.2	14.5	2
1	11.0	8.5	2.0	0
35	13.0	8.6	9.0	2
29	13.0	9.3	17.0	2
26	12.0	10.2	13.0	1
8	14.0	10.0	3.0	0
38	13.0	8.1	10.0	2
9	13.0	8.7	7.0	0
31	15.0	10.0	2.0	2
33	12.0	9.5	7.0	2
37	12.0	10.0	11.5	2
14	11.0	8.5	3.0	1
17	14.0	7.3	2.0	1
19	12.0	9.0	5.0	1
28	13.0	8.4	3.0	2
30	12.0	9.7	2.0	2
20	12.0	9.8	3.0	1
39	13.0	8.8	14.0	2

[6]:
```
f       p    figure
        f    add_subplot 1       projection   '3d
    scatter   seed2_d '          seed2_d   '         seed2_d '
                  seed2_d 't
p    xlabe   r'$X_0[mm]$'
p    ylabe   r'$Y_0[mm]$'
p    title   r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$'
p    gri
```

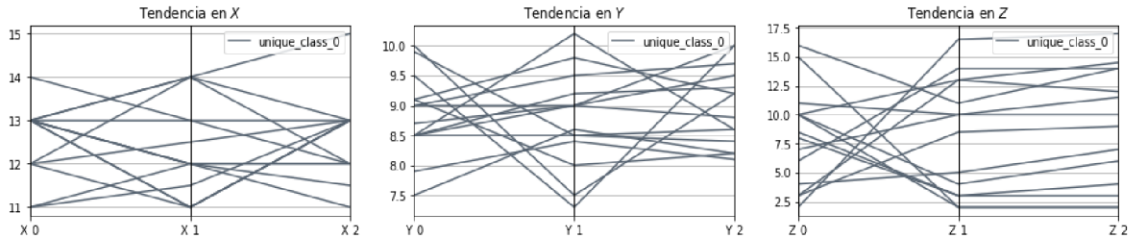


```
s    scatterplot  seed2_d '      seed2_d '      h   seed2_d 't
                  palette  s   color_palette  "tab10   n_color
s    set_palette "tab10
p    xlabe r'$X[mm]
p    ylabe r'$Y[mm]
p    title   r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$'
p    gr
```




```
delta1    seed2_d l   seed2_d 't          '    value seed2_d l    seed2_d
delta2    seed2_d l   seed2_d 't          '    value seed2_d l    seed2_d
```

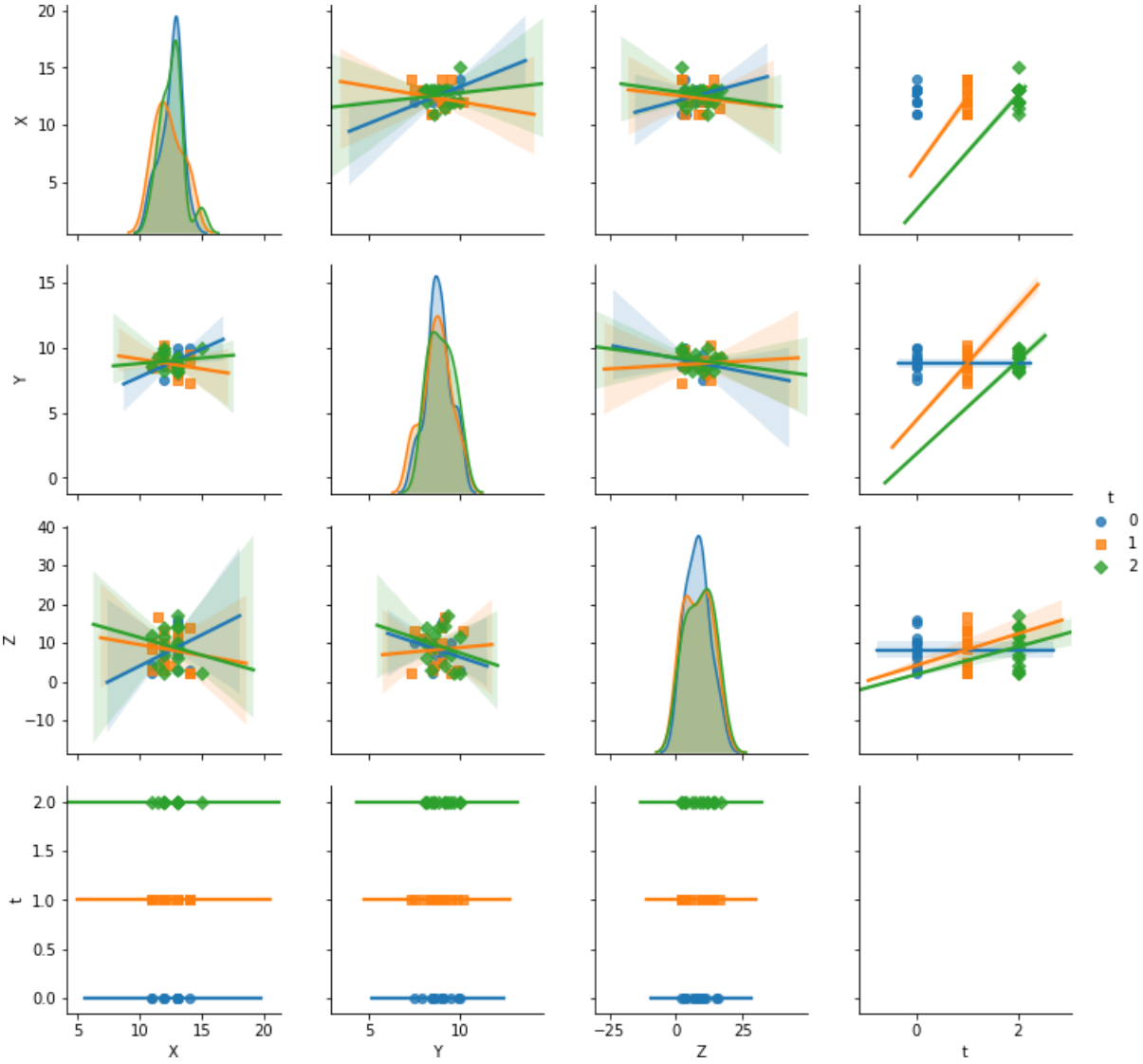
[9]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
pd.plotting.parallel_coordinates(
        df[['Z_0','Z_1','Z_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Z$')
plt.show()
```



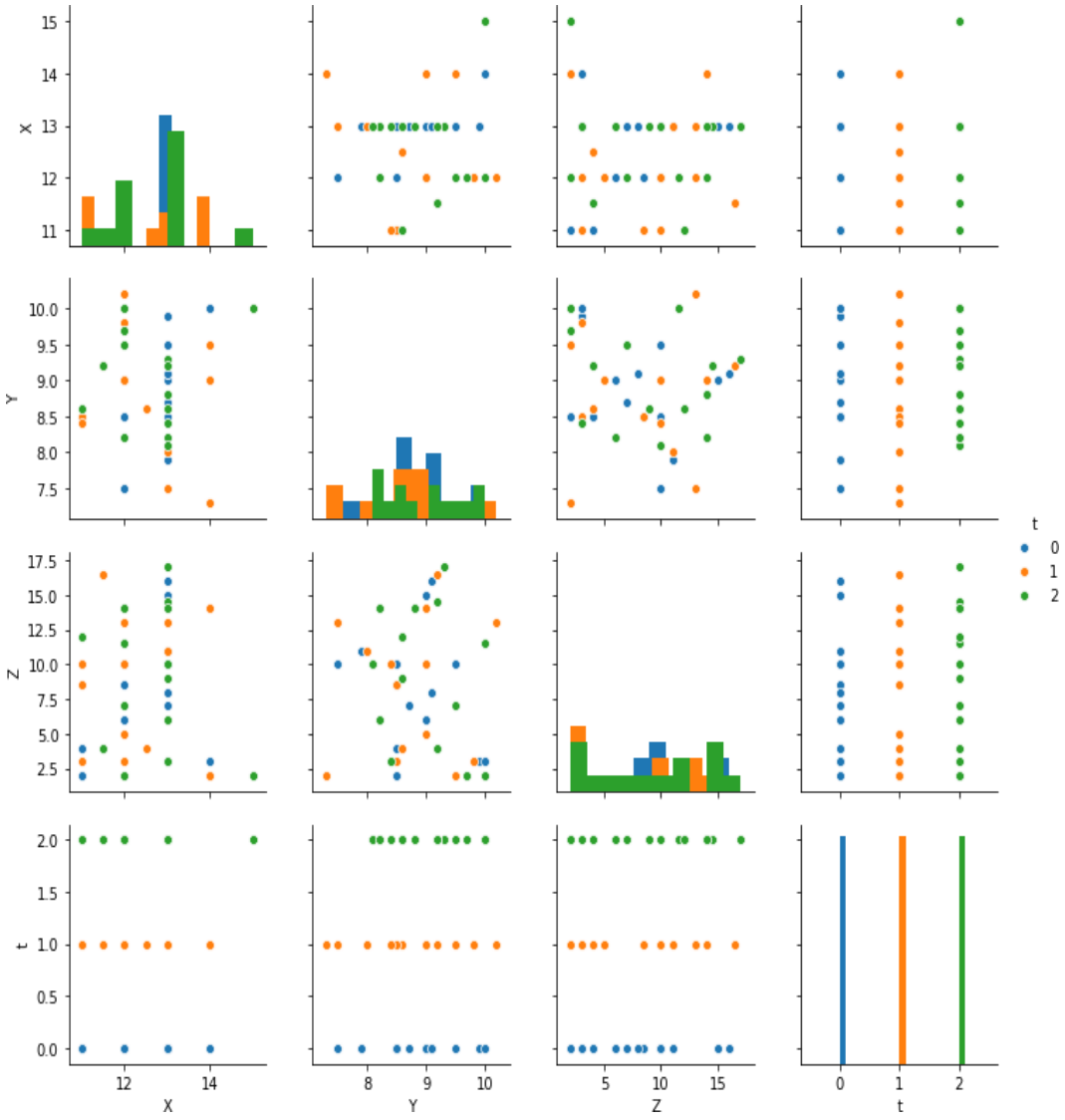
[17]:
```
s    pairplot  seed2_d  h   't    ki   "reg   marker  "    "    "
p    sh
```

C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487: RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

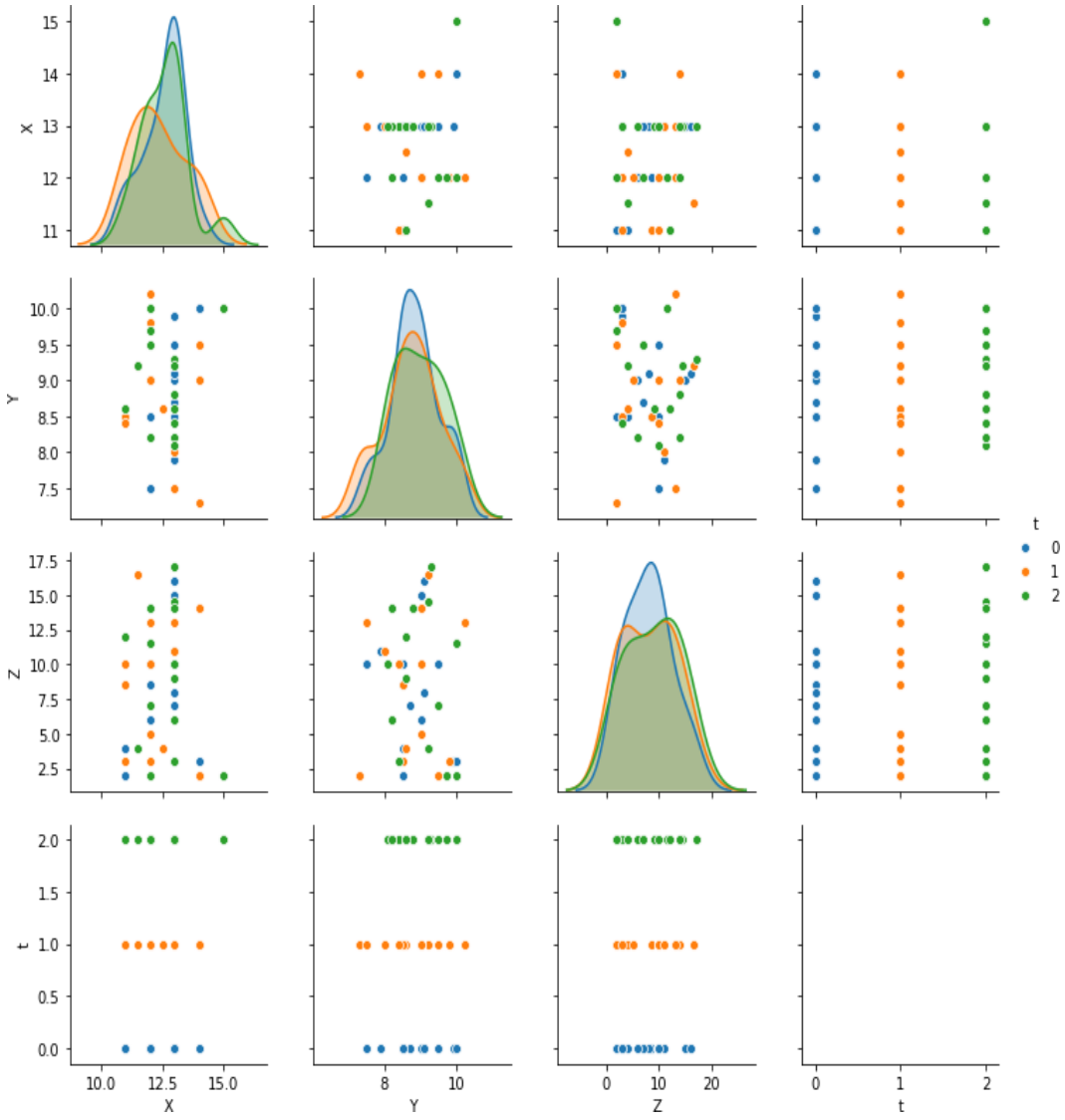
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



[11]:
```
sns.pairplot(seed2_df, hue='t', diag_kind='hist')
plt.show()
```

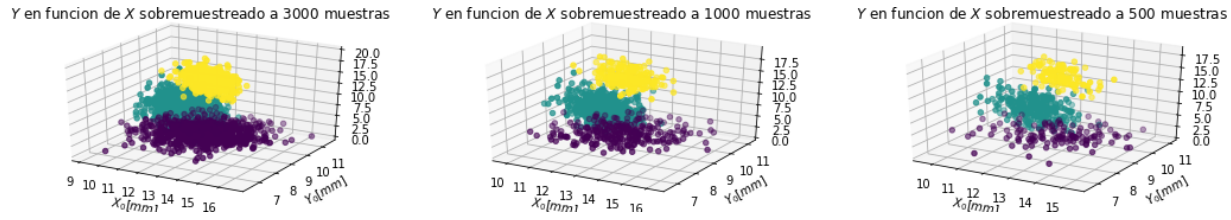


[12]:
```
s   pairplot  seed2_d   h   't
p   sh
```



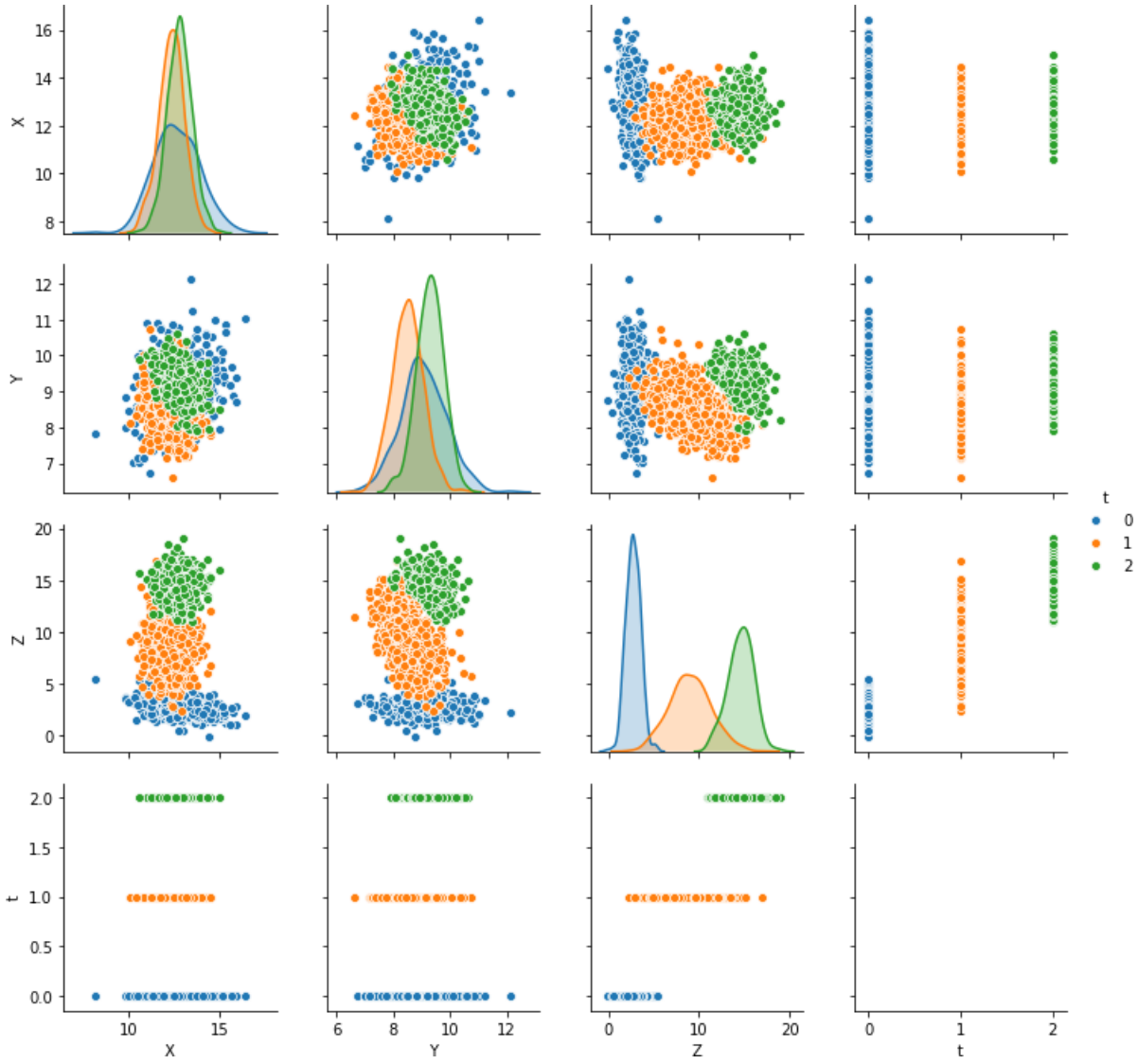
```
[15]: from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
   n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(3),     init_params="kmeans", max_iter=100,
random_state=2).fit(seed2_df[['X', 'Y',
In [16]: X_s, t_s = dpgmm.sample(n_samples=3000) X_s
Out[16]: array([[11.76464939,  9.36520096,  5.02096069],       [11.50504237,  8.57813838,
6.21908969],
       [13.96134929,  8.11519571, 10.3819972 ],       ...,
       [11.6997626 , 9.79296316, 14.1509871 ],
       [12.61312577,  9.03639821, 16.97353843],       [13.12320351, 9.34360481, 13.9399642 ]])
```

```
[23] : fig = plt.figure(figsize=(18,3)) ax = fig.add_subplot(131, projection='3d') X_s, t_s =
dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
ax = fig.add_subplot(132, projection='3d') X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
ax = fig.add_subplot(133, projection='3d') X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],          c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```




```
[24] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s,
columns=['X', 'Y', 'Z']),                    pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
```

```
sns.pairplot(seed2over_df, hue='t') plt.show()
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:
RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta
* nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning:
invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2
```



[1]:
```
 matplotlib inline
impo  nu
impo  pand
impo  seabor      s
fr  matplotlib  impo  pyplo      p
```

[2]:
```
fr  sklear  ba  impo  BaseEstimator  TransformerMixin

cla  DataFrameImputerTransformerMixin

    d  _init___sel
        """Impute missing values.

        Columns of dtype object are imputed with the most frequent value
        in column.

        Columns of other types are imputed with mean of column.

        """

        d  f  sel          No
            sel  fil          Serie          value_counts(  ind
                          dty          dty  '  el        m      f
                ind      colum

            retu  sel

        d  transform  sel          No

            retu     fillna  sel  fil
```

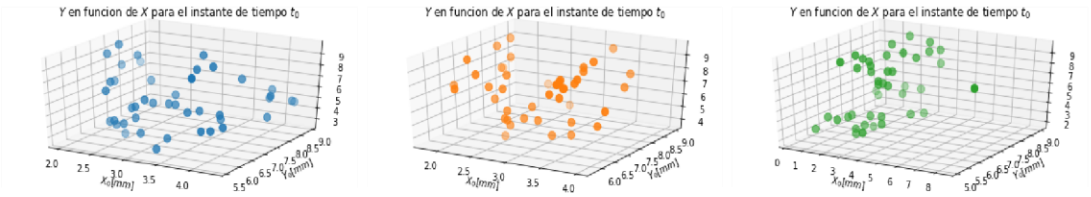
Out[4]:

[4]:
```
impo  pand
d2          read_excel '3dcristina.xlsx'
d2  replac          n      inplac  Tr
d2      DataFrameImpute(r  fit_transform  d2
d2
```

X	Y	Z	X	Y	Z	X	Y	Z
2.8	6.3	4.7	2.6	6.4	7.7	2.7	6.4	3.8
2.8	6.3	5.7	3.3	6.3	7.9	2.4	6.0	2.8
3.2	6.8	5.7	2.4	6.4	7.7	2.5	6.3	2.9
3.5	6.5	5.6	3.4	6.4	7.6	2.4	6.0	2.3
3.1	6.8	5.7	3.3	6.8	7.9	2.1	7.0	1.9

3.6	6.7	5.6	2.5	6.8	8.3	1.7	6.4	3.6	
2.9	6.7	5.7	3.3	6.7	8.6	2.0	6.0	3.9	
2.7	6.4	5.4	2.3	6.2	8.7	1.0	5.4	3.0	
3.0	7.3	5.9	3.4	6.3	8.7	3.8	6.0	3.7	
3.9	6.7	5.5	3.5	6.3	8.5	3.3	6.4	3.9	
2.9	6.4	6.3	3.4	6.4	8.8	3.9	5.1	3.8	
2.5	7.6	6.5	3.5	6.8	8.7	3.9	7.8	3.0	
2.4	6.2	6.9	2.2	7.7	8.7	3.9	7.2	3.5	
2.3	6.6	7.2	3.4	7.3	9.1	0.3	8.6	3.5	
2.0	7.7	7.4	3.5	7.4	9.5	0.8	8.3	3.5	
3.8	7.8	7.5	2.1	6.4	9.5	1.3	5.7	4.0	
2.0	7.2	8.9	2.0	7.7	9.4	1.8	8.7	4.3	
3.6	6.4	8.8	2.0	8.2	9.6	2.2	5.4	4.3	
3.6	7.2	9.0	2.7	6.2	5.7	2.8	8.6	4.4	
3.0	6.0	3.8	3.3	6.4	4.7	2.8	6.9	6.6	
3.4	6.0	3.7	3.5	6.6	4.7	3.3	6.7	6.7	
2.5	6.3	3.8	2.8	8.4	4.8	3.4	9.0	6.8	
3.4	5.6	3.2	2.7	6.4	5.0	2.8	6.7	6.7	
2.4	6.3	4.0	3.1	6.4	5.7	2.8	6.7	7.0	
2.7	6.1	4.3	2.9	6.4	5.7	2.3	6.8	7.7	
3.9	6.0	4.8	3.5	6.6	5.7	3.3	6.7	8.1	
3.8	7.0	4.7	3.9	6.5	5.8	2.4	6.7	8.4	
4.0	6.1	4.8	2.7	6.3	5.5	2.2	6.8	8.7	
2.6	6.0	4.6	2.9	6.7	6.3	2.2	6.3	8.5	
4.1	6.3	5.0	2.4	8.6	6.0	2.0	6.7	9.0	
4.2	8.9	4.6	2.7	6.3	6.2	2.2	7.8	9.5	
	X_0	**Y_0**	**Z_0**	**X_1**	**Y_1**	**Z_1**	**X_2**	**Y_2**	**Z_2**
4.3	7.5	4.8	2.4	6.6	6.6	3.5	8.4	9.5	
4.3	8.5	5.2	4.0	6.5	7.0	2.2	8.6	9.6	
4.0	8.4	5.3	3.8	7.8	7.5	2.1	5.7	9.6	
4.0	8.5	6.0	1.8	6.6	7.2	3.4	7.3	9.4	
4.3	7.6	6.6	1.8	8.4	7.0	2.0	6.4	9.6	
2.1	7.2	6.5	2.1	7.6	7.9	2.4	7.4	8.8	
2.0	8.4	7.0	3.2	9.0	9.0	3.3	6.9	7.9	
2.1	7.4	9.6	3.5	6.7	8.5	3.6	8.8	8.4	
3.3	7.6	9.4	3.4	7.3	8.8	3.6	7.7	8.6	
2.3	7.6	8.6	3.7	8.3	8.0	3.7	8.2	8.8	
3.4	7.1	8.7	2.2	5.7	8.0	8.3	6.2	8.3	

In [5]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(figsize=(18,3))
ax = fig.add_subplot(131, projection='3d')
ax.scatter(d2df['X_0'], d2df['Y_0'], d2df['Z_0'],
           color=sns.color_palette("tab10", n_colors=10)[0],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(132, projection='3d')
ax.scatter(d2df['X_1'], d2df['Y_1'], d2df['Z_1'],
           color=sns.color_palette("tab10", n_colors=10)[1],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()

ax = fig.add_subplot(133, projection='3d')
ax.scatter(d2df['X_2'], d2df['Y_2'], d2df['Z_2'],
           color=sns.color_palette("tab10", n_colors=10)[2],
           s=70)
plt.xlabel(r'$X_0[mm]$')
plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$')
plt.grid()
plt.tight_layout()
plt.show()
```



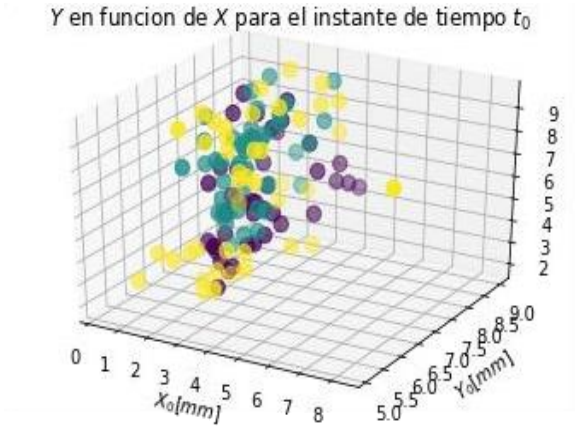
In [7]:
```
d2df['t_0'] = np.zeros(len(d2df))
d2df['t_1'] = np.ones(len(d2df))
d2df['t_2'] = np.ones(len(d2df))*2
seed2 = np.vstack((d2df[['X_0', 'Y_0', 'Z_0', 't_0']].values,
                   d2df[['X_1', 'Y_1', 'Z_1', 't_1']].values,
                   d2df[['X_2', 'Y_2', 'Z_2', 't_2']].values))
seed2_df = pd.DataFrame(seed2, columns=['X', 'Y', 'Z', 't'])
seed2_df['t'] = seed2_df['t'].astype(int)
print('Dimension: ', seed2_df.shape)
seed2_df.sample(30)
```

```
Dimension:  (129, 4)
```

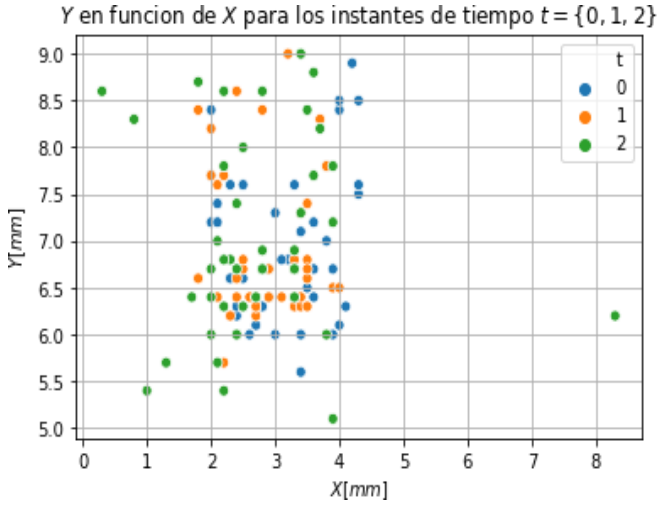
Out[7]:

	X	Y	Z	t
78	1.8	6.6	7.2	1
107	3.3	6.7	6.7	2
75	2.4	6.6	6.6	1
27	3.8	7.0	4.7	0
7	2.9	6.7	5.7	0
60	2.0	7.7	9.4	1
32	4.3	7.5	4.8	0
26	3.9	6.0	4.8	0
68	2.9	6.4	5.7	1
112	3.3	6.7	8.1	2
87	2.7	6.4	3.8	2
21	3.4	6.0	3.7	0
120	2.1	5.7	9.6	2
59	2.1	6.4	9.5	1
31	4.2	8.9	4.6	0
23	3.4	5.6	3.2	0
81	3.2	9.0	9.0	1
80	2.1	7.6	7.9	1
128	8.3	6.2	8.3	2
94	1.0	5.4	3.0	2
118	3.5	8.4	9.5	2
40	3.3	7.6	9.4	0
55	3.5	6.8	8.7	1
84	3.7	8.3	8.0	1
105	2.8	8.6	4.4	2
100	0.3	8.6	3.5	2
83	3.4	7.3	8.8	1

```
f       p    figur
       f    add_subplo 1      projection '3
    scatter  seed2_   '       seed2_   '       seed2_   '
                seed2_   '
p    xlab    r'$X_0[mm]
p    ylab    r'$Y_0[mm]
p    titl    r'$Y$ en funcion de $X$ para el instante de tiempo $t_0$'
p    gr
```

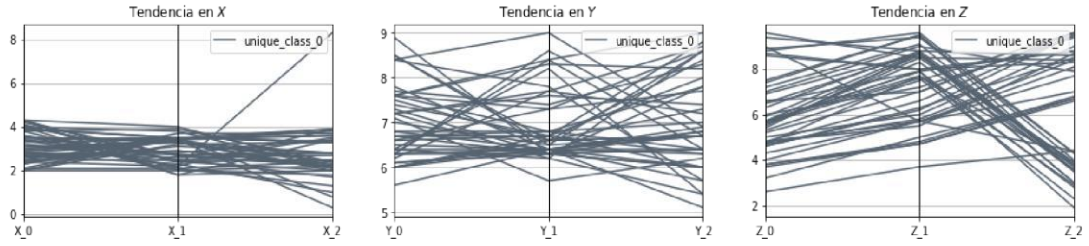



```
        s    scatterplot  seed2_   '       seed2_   '       h    seed2_   '
                          palett   s    color_palette  "tab1     n_color
s    set_palette "tab1
p    xlab    r'$X[mm]
p    ylab    r'$Y[mm]
p    titl    r'$Y$ en funcion de $X$ para los instantes de tiempo $t=\{0, 1, 2\}$'
p    gr
```



In [10]:
```
delta10 = seed2_df.loc[seed2_df['t'] == 1]['Y'].values-seed2_df.loc[seed2_df['t'
delta21 = seed2_df.loc[seed2_df['t'] == 2]['Y'].values-seed2_df.loc[seed2_df['t'
```

In [12]:
```
df = d2df.copy()
df = pd.concat([df, pd.DataFrame(np.zeros(len(d2df)), columns=['class'])], axis=
df.replace(0, 'unique_class_0', inplace=True)
df_0 = seed2_df.loc[seed2_df['t'] == 0].reset_index(drop=True)
df_1 = seed2_df.loc[seed2_df['t'] == 1].reset_index(drop=True)
df_2 = seed2_df.loc[seed2_df['t'] == 2].reset_index(drop=True)

plt.figure(figsize=(18,3))
plt.subplot(131)
pd.plotting.parallel_coordinates(
        df[['X_0','X_1','X_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $X$')


plt.subplot(132)
pd.plotting.parallel_coordinates(
        df[['Y_0','Y_1','Y_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Y$')

plt.subplot(133)
pd.plotting.parallel_coordinates(
        df[['Z_0','Z_1','Z_2', 'class']], 'class',
        color=('#556270'))
plt.title(r'Tendencia en $Z$')
plt.show()
```

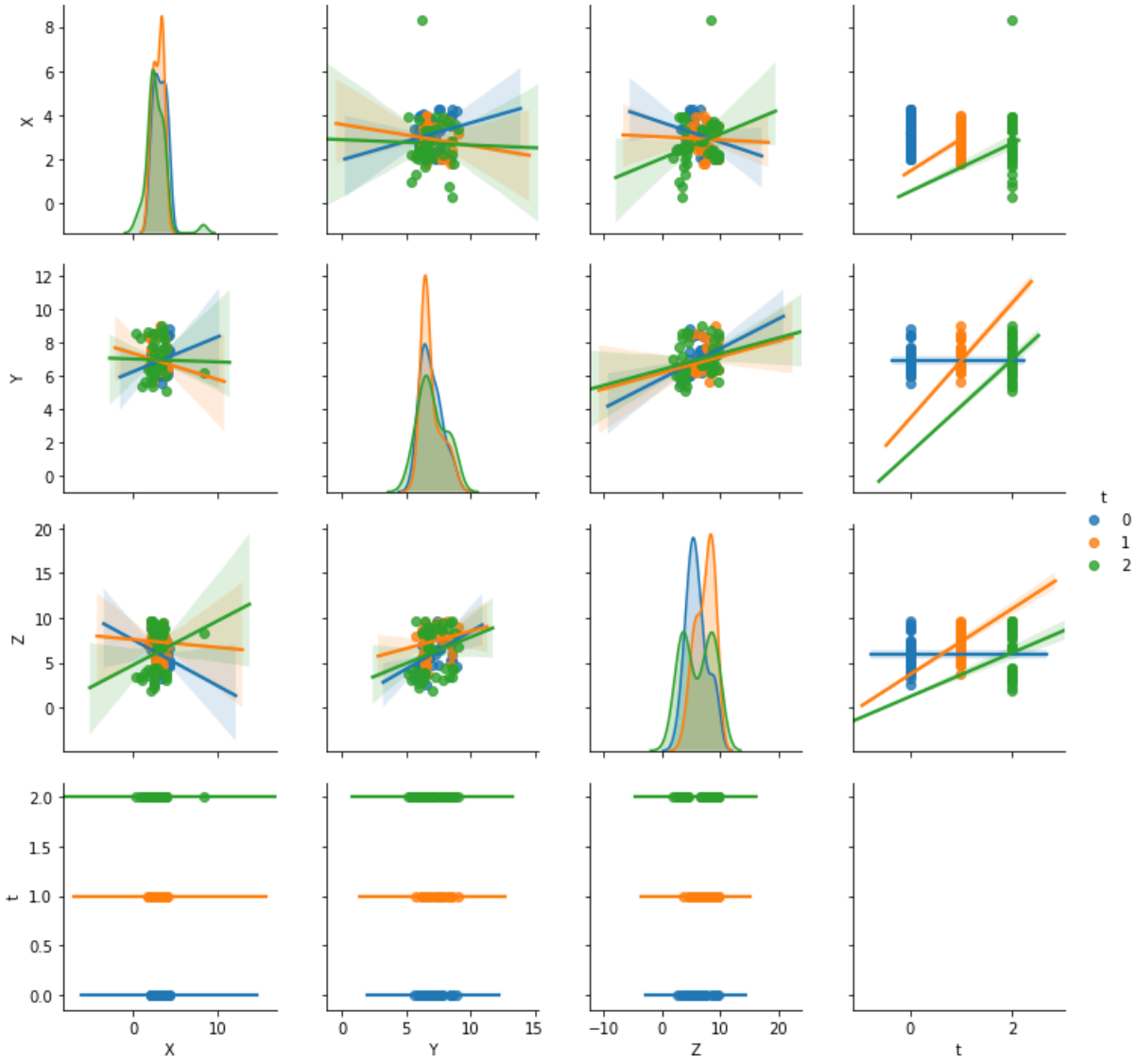


C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:

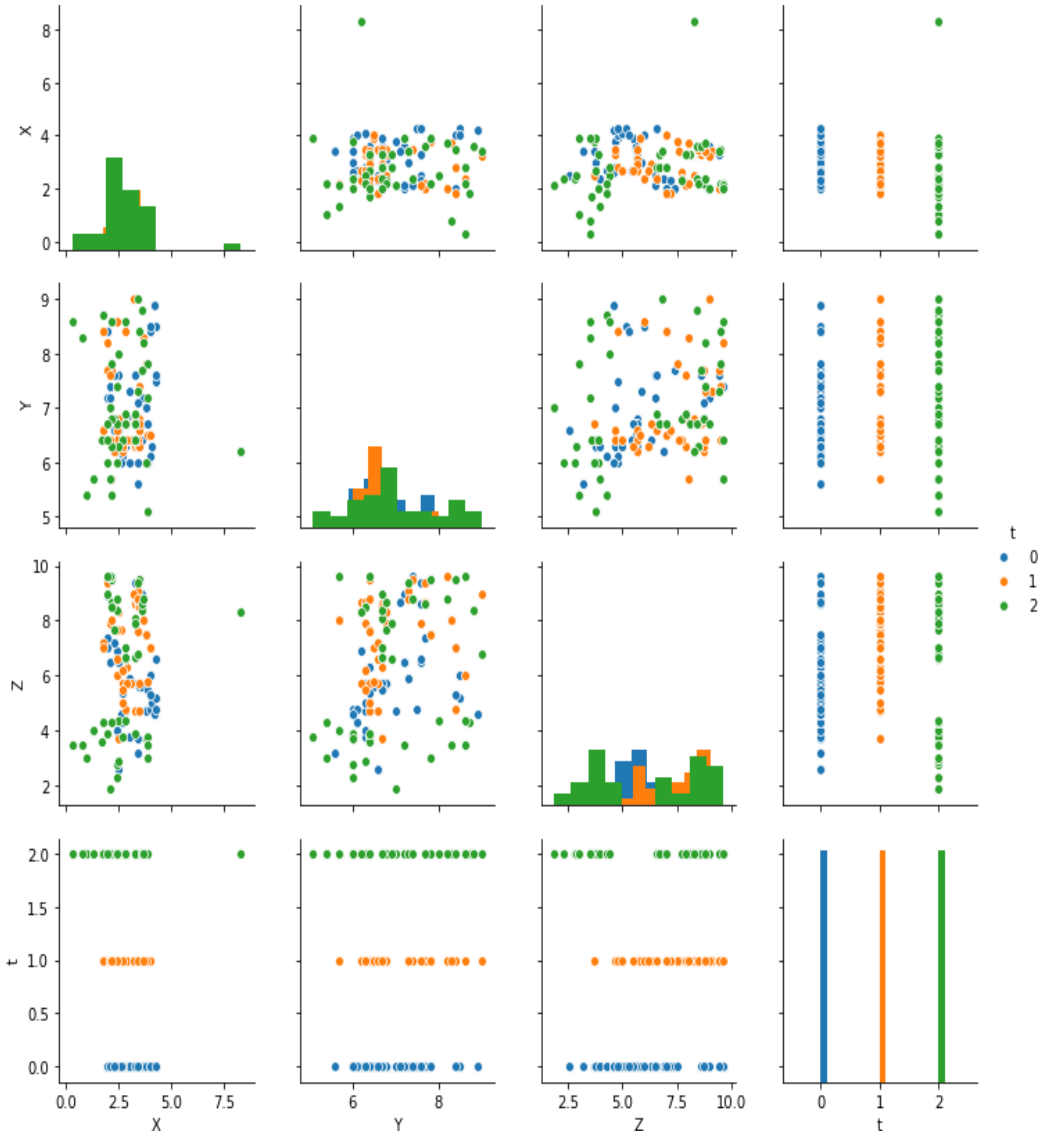
```
[13]: s    pairplot  seed2_d   h    't     ki    "reg
      p    sh
```

RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)

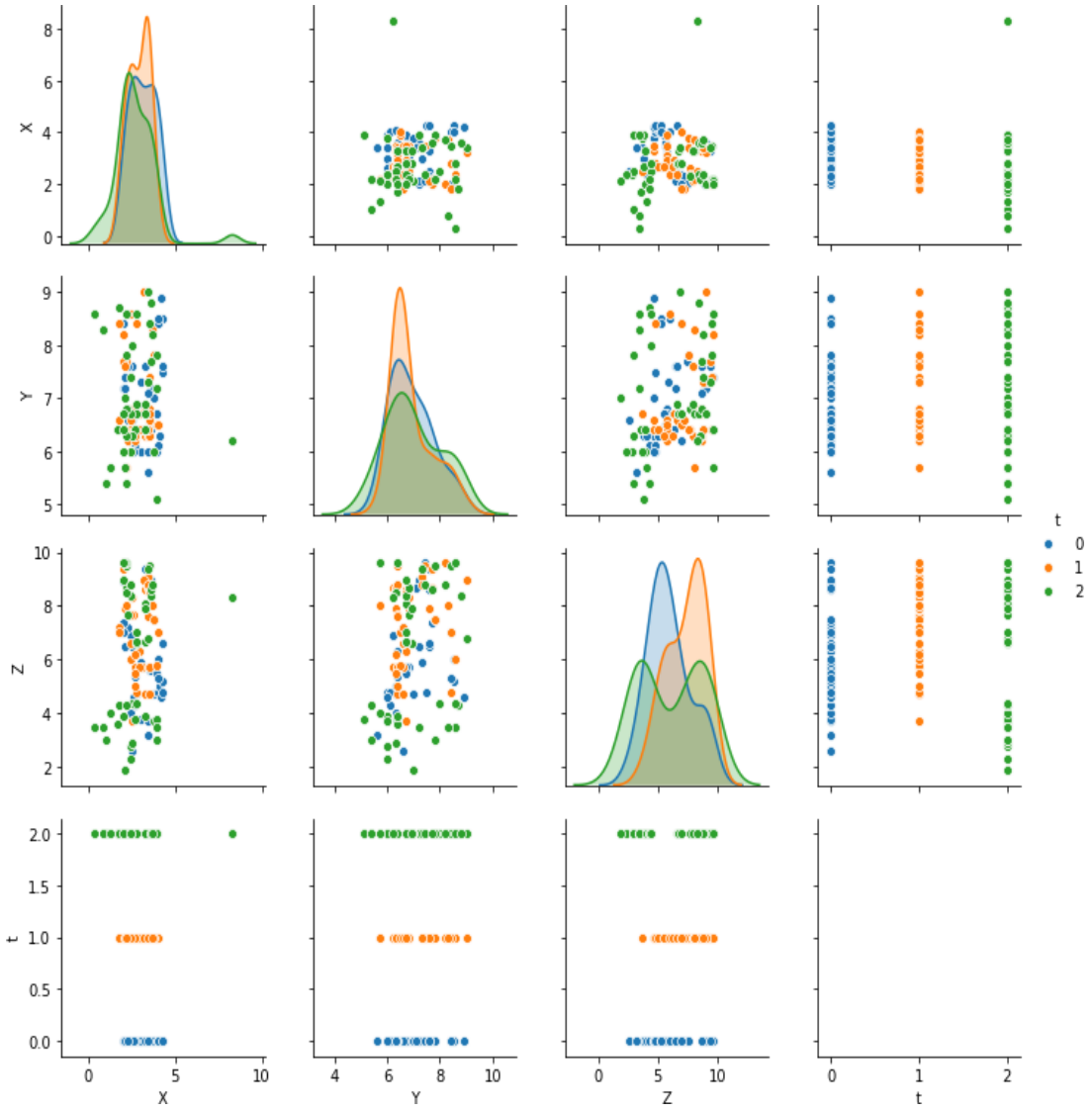
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning: invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2



```
s    pairplot seed2_      h    't      diag_kin 'hist'
p    sh
```

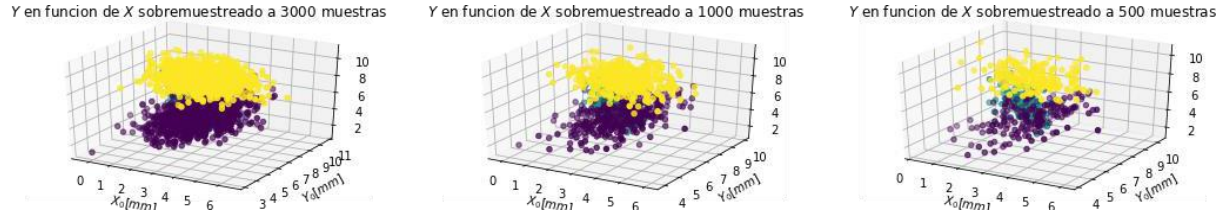


[15]:
```
s    pairplot    seed2_d    h    't
p    sh
```



```
[16]: from sklearn import mixture
dpgmm = mixture.BayesianGaussianMixture(
   n_components=3, covariance_type='full', weight_concentration_prior=1e+2,
weight_concentration_prior_type='dirichlet_process',     mean_precision_prior=1e-2,
covariance_prior=1e0 * np.eye(3),     init_params="kmeans", max_iter=100,
random_state=2).fit(seed2_df[['X', 'Y',
In [17]: X_s, t_s = dpgmm.sample(n_samples=3000) X_s
Out[17]: array([[3.6117502 , 5.67766287, 4.83387898],       [3.76215784, 5.88660016, 5.5336117 ],
       [2.63117037, 7.87687037, 6.08411859],       ...,
       [2.41852165, 6.03659038, 8.34312387],
       [4.43510864, 6.65100071, 7.98671698],       [2.81523679, 7.51763011, 9.06867099]])
```

```
[18] : fig = plt.figure(figsize=(18,3)) ax = fig.add_subplot(131, projection='3d') X_s, t_s =
dpgmm.sample(n_samples=3000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_3000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 3000 muestras') plt.grid()
ax = fig.add_subplot(132, projection='3d') X_s, t_s = dpgmm.sample(n_samples=1000)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_1000_muestras.csv') ax.scatter(seed2over_df['X'], seed2over_df['Y'],
seed2over_df['Z'],          c=seed2over_df['t'], s=20) plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 1000 muestras') plt.grid()
ax = fig.add_subplot(133, projection='3d') X_s, t_s = dpgmm.sample(n_samples=500)
seed2over_df = pd.concat([pd.DataFrame(X_s, columns=['X', 'Y', 'Z']),
pd.DataFrame(t_s, columns=['t'])], axis=1) seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
seed2over_df.to_csv('cristales_3d_500_muestras.csv')
ax.scatter(seed2over_df['X'], seed2over_df['Y'], seed2over_df['Z'],          c=seed2over_df['t'], s=20)
plt.xlabel(r'$X_0[mm]$') plt.ylabel(r'$Y_0[mm]$')
plt.title(r'$Y$ en funcion de $X$ sobremuestreado a 500 muestras') plt.grid()
```



```
[19] : X_s, t_s = dpgmm.sample(n_samples=1500) seed2over_df = pd.concat([pd.DataFrame(X_s,
columns=['X', 'Y', 'Z']),                    pd.DataFrame(t_s, columns=['t'])], axis=1)
seed2over_df['t'].replace([0, 1], [1, 0], inplace=True)
sns.pairplot(seed2over_df, hue='t') plt.show()
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:487:
RuntimeWarning: invalid value encountered in true_divide binned = fast_linbin(X, a, b, gridsize) / (delta
* nobs)
C:\Users\ufps\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.p y:34: RuntimeWarning:
invalid value encountered in double_scalars FAC1 = 2*(np.pi*bw/RANGE)**2
```