Die Funktionsweise von PythonSoftIOC wurde auch für das Projekt getestet. Dazu wurden zuerst vom Python 3.9 Interpreter aus eine Reihe von PVs erzeugt und getestet. Zum Testen Software HZB. Darauf folgend wurde das gleichzeitige Laufen von EPICS über PythonSoftIOC und im Zusammenhang mit der Kamera getestet um sicher zu stellen das sowohl die Kamera als auch die EPICS Schnittstelle gleichzeitig ausgelesen werden kann. Als auch das erfolgreich umgesetzt werden konnte, stand der Verwendung von PythonSoftIOC nicht mehr im Wege.

\section{Zusammenführen der Umsetzungsschritte}

Das Anbinden der Kamera, Datenasuwertung und … konnten erfolgreich getestet werden. Im nächsten Schritt wurden diese unabhängigen Programmteile zusammengefügt. Durch das Einarbeiten in die jeweiligen Themengebiete mit einem hohen zeitlichen Aufwand, war das zusammen setzten der Teil Schritte bis zum ersten Prototyp mit einem geringerem Zeitaufwand verbunden. Vor dem eigentlichen Code zusammen fügen wurde zuerst eine Architektur für den Code entworfen. Diese sollte, wie in den Anforderungen (siehe bla) spezifiziert, besonders anpassungsfähig und erweiterbar sein.

Das Festlegen auf die Programmiersprache Python bestimmt verwendete Programmierparadigma – objektorientierte Programmierung. Beim objektorientierten Programmieren …

Um den Code für die Lösung eine Form zu geben wurde zu Beginn ein Klassendiagramm entworfen und im zweiten Schritt sich mit den Programmstrukturen für ... auseinander gesetzt.

Bevor die Klassen vollständig implementiert wurden Entwurf und Struktur an vereinfachten Klassen getestet danach erst im Detail implementiert. Bei der Detailimplemtierung konnten große Teile des Codes von den vorherigen Umsetzungsschritte übernommen werden. Bei der Implemtierung von Bibliotheks besonderheiten, war die Doumentation vorher in Jypiter Notebooks besonders hilfreich.

\subsection{Entwurf Klassendiagramm}

Aus den an Abb. x,y erklärten Anforderungen wurden eine Reihe von Klassen für die Implementierung abgeleitet.(siehe Abb) Die "CamIOC" steht für "Camera Input/Output Controler", angelehn

Diese alternative Lösung für den Informationsaustausch birgt jedoch die Gefahr, dass der Zugriff auf die Klasse "CamDatEps" unerwünscht ausgenutzt wird. Alle Bemühungen für Unabhängigkeit wären dahin, wenn z.B. im "Epics" Code über den "CamDatEps" ohne Vermittlungsfunktion direkt auf "DataAnalyzer" Werte zugriffen werden würde. Dann könnten Änderungen im „DataAnalyzer“ umbemerkt Änderungen in „Epics“ hervor rufen. Um das zu verhindern wurde sich das Konzept der Datenkapselung bei objektorientierter Programmierung zu Nütze gemacht. Bei Datenkapselung geht es darum ... retrict access. Die Idee ist durch schon im Code festgehaltende Regeln, spätere Falschimplementationen zu vermeiden. In anderen Programmiersprachen wie z.B. Java .. über .. gelöst. Ist eine Funktion als private eingestuft, nur von dem Objekt ausgelesen werden angehört. Protected... In Python wird diese Einstufung über Namen der Atribute und Funktionen festgelegt. Protected ... markiert, Private ...markiert. Protected kaum Auswirkung, auch Warnung für Programmierer private erstmal nicht mehr unter dem Namen zu finden, kann aber mit magln umgehen werden. Trotzdem wichtiges Konzept und auch für hier von Bedeutung. Aus Abb xy zu sehen IOC Zugriff auf CamDatEps aber die CamDatEpz Objekte nicht gegenseitig beienflussen.

Nochmal warum entscheidung trotz Programierumgebung änderung erleichtern und über Tools Zusammenhänge klar machen, nochmal bezug ausbaubarkeit

Ein weiteres Konzept der oop, das für die Entwicklung des Klassendiagramms von Bedeutung war, ist Polymorphie. Wie in Abschnitt xy deutlich wird. Gibt es für die Anbindung der Kamera verschiedene Schnittstellen, wie die Vimba Python API und Harvester, und damit verschiedene Möglichkeiten für die Bilderfassung. Außerdem wäre es von Nutzen wenn die Bilderfassung durch das Laden bereits aufgenommener Aufnahmen für Testzwecke simuliert werden könnte. Ein Lösungsansatz wurde hierfür mit dem Konzept der Polymorphie entworfen. P beschreibt... In diesem Fall gibt es eine Klasse "ImageAquirer" deren funktional wichtigen Methoden erst von ihren ....(siehe Abb) So wird in der Klasse "ImageAquirerFile" das laden eines Bildes durch das Laden eines Bildfiles implementiert. Bei dieser die Art Umsetzung für den "ImageAquirer" beliebig ausgetauscht werden .

\section{Datenanalyse}

Die von der Kamera produzierten Daten, können zwar den Elektronenstrahl visualisieren, eine genauere Analyse der Daten, kann weitere Details hervorbringen. Für die Steuerung von BESSY II werden bisher nicht nur die Positionswerte des Elektronenstrahls auf dem Kamerabild in Betracht gezogen, sondern auch die Verteilung des Elektronenstrahls. Um die bisherige Umsetzung mit LabView ablösen zu können soll auch dieser Teil in Python implementiert werden. Im Folgenden wird erläutert wie aus den Bilddaten spezifische Informationen extrahiert werden, die dann an das Prozessleitsystem weitergegeben werden können.

Die entworfenen Datenanyle Algorithmen wurden mit Bilddaten getestet. Zu Beginn beruhten diese Daten auf den in (blla ) beschriebenen Aufnahmen. Die Verteilung um den hellen Punkt, der den Elektronenstrahl simuliert, konnte jedoch nicht zufrieden stellend simuliert werden. Daten während des Betriebes von BESSY II aufzunehmen war auch nicht möglich, da diese Arbeit/Projekt in einer Zeit entstanden ist, in der BESSY II in Sommerpause, also nicht in Betrieb war. \\

Glücklicherweise, gibt es in Adlershof einen MLS, …, Bei der MLS gibt es einen Vorbeschleuniger an dem entsprechend der Möglichkeit x in Abschnitt x.y mit Hilfe von Flouressenzschirmen mit Hilf evon CCD\_Kameras Daten erhoben werden. Während eines Testlaufes war es möglich die entsprechende Kamera mit der Vimba Python API anzusteuern und auszulesen. Der auf den Bilddaten sichtbare Lichtfleck für den Elektronenstrahl ist gut auswertbar.

Die Bilddaten liegen in Form eine np.arrays vor. Np. Array Beschreibung. Shape Bild

\label{Datenformat}

Fit analyse im groben Erklären. Fit Model parameter

was soll am Ende raus kommen (Fit Parameter)

welche schritte durchlaufen Performanze wichtig zu analysierende Daten aufbereiten (Fit um so besser desto weniger Daten)

Basically aufbereiten der Daten, und dann Fit.

1. Daten Aufbereitung

Die Kamera kann bis zu 33 Bilder pro Sekunde aufnehmen, d. h. es müssten … Pixel pro Sekunde ausgewertet werden. Um die Auswertungsalgorithmen zu beschleunigen ist daher ein wichtiger gleich am Anfang der Analyse die Menge der Daten zu reduzieren. Um Informationen über den Lichtstrahl zu erhalten, reicht es sich nur den Bereich um einen der Elektronenstrahl verdeutlichenden hellen Punkte anzuschauen. Alle anderen schwarzen Pixel sind für die Analyse nicht relevant. Den richtigen Bereich auszuwählen erfordert einen hohen mathematischen Aufwand auf Basis physikalischer Überlegungen. In Absprache mit dem Betreuer der Arbeit Hr. Prof. Dr. Rehm, wurde sich deshalb dafür entschieden, dass dieser Bereich manuell von Nutzer:innen festgelegt werden kann. Der erste Schritt in er Datenanyse ist also dementsprechend das große Bild in einen für die Analyse relevanten Bereich zuzuschneiden. Ein solcher Bereich wird in der Bildanalyse auch als „region of interest“ bezeichnet.

In xy wurde bereits erwähnt, dass Hotpixel bei der Auswertung mit beachtet werden müssen. Diese einzelnen Pixel mit maximal Werten sind auf die technischen … der Kamera zurück zu führen und verfälschen das Ergebnis. Es gibt verschiedene Algorithmen um Ausreißer in Datenmengen zu eliminieren. In diesem Fall wird der median filter verwendet. Erklärung, warum Wahl, zeitliche Aspekte

Die Menge der Daten wird bereits durch eine manuell festgelegte roi verkleinert. Es ist jedoch nicht sicher, dass der Elektronenstrahl immer genau an der gleichen Stelle in der Bildaufnahme abgebildet ist. Es gab Überlegungen, die roi immer um die Elektronenstrahlabbildung mitlaufen zu lassen. Der Bereich würde sich also an die Position des Strahls anpassen. Diese Variante wurde jedoch abgelehnt, da es dann außer zu Beginn nie eine Möglichkeit für Nutzer:innen gibt die roi zu beinflussen. Stattdessen wurde sich mit Prof. Dr. Rehm darauf verständigt, die roi weiterhin manuell zu bestimmen, dann aber innerhalb der roi den Strahl algorithmisch zu finden und nocheinmal um den Strahl herum einen Bereich zu definieren, der weiter verarbeitet wird. Sollte sich der Elektronenstrahl an den Rand der roi bewegen, könnte ein Hinweis an den/die Nutzer:in geschickt werden, sodass diese die roi aktiv verbessert. Dadurch bleicht eine gewisse Flexibilität für die Position des Strahls erhalten, schließlich kann sich der Strahl innerhalb der roi bewegen. Sollte die Bewegung zu stark sein, ist das ein wichtiger Hinweis für die Kontrolle des Strahls und insofern vorteilhaft, wenn dann die roi aktiv verschoben werden muss. Außerdem ist es noch einmal möglich, die bereits kleinere Menge an Daten weiter zu verkleinern.

Clustering

Um algorithmisch zu bestimmten in wo sich die die Abbildung des Elektronenstrahls in der roi befindet ist ein klassisches Problem der Datenanalyse. Ähnliche Datensätze in ein Cluster sortieren… BAnwendungsbeispiele Clustering. Nicht nur Clsuter finden auch später neue Werte in Cluster ein ordnen. Nicht für diesen Problemfall interessant. Es gibt eine breite Auswahl an Clustering Algorithmen die sich vor allem darin unterscheiden, wie sie bestimmten ob zwei Datensätze einander ähnlich sehen. In diesem Fall besteht ein Datensatz aus einem Helligkeitswert und eine x- und y- Position im Bild. Je ähnlicher die Helligkeit und so näher sie aneinander liegen, desto ähnlicher die zwei Datensätze. Aus der shape… geht hervor, dass bisher nur die Helligkeit in den Bilddaten zu finden ist. Für jeden Punkt in den Bilddaten muss zusätzlich genauso große Datenmengen erzeugt werden, die die Positionen der Bildwerte festhalten. Um also die Bildwerte in Cluster zu fügen werden die Daten mit einem Medianfilter aufbereitet um Verfäschungen zu vermeiden, dann werden dazu gehörige Positionswerte erzeugt und daraufhin werden die Daten an einen Clustering Algorithmus weitergegen. Für Clustering Algorithmen ist die skilearn Bibliothek eine besonders „gute“ Wahl. Sie liefert eine Vielzahl ausgefeilter Clustering Algorithmen. Das Hauptkriterium für die Auswahl der Algorithmus war die Laufzeit. Da das Ergebnis des Algorithmus schon bekannt ist, muss die Abstandsberechnung nicht besonders tiefgreifend und komplex sein. Es reicht eine einfache Abstandsberechnung die verlässlich das eine Cluster im Bild findet. Deswegen wurde sich für den KMeans Algorithmus entschieden. Laut skilearn … Zuverlässig, Ergbnis Array clustern, Bild, Schon beim ersten Testlauf zeichnete sich ab, dass KMeans zwar verhältnismäßig schnell ist, wenn man ihn mit anderen Clusteringalgorithmen vergleicht, aber schon bei einer kleinen Datenmenge bis zu mehreren Sekunden gebraucht hat um zu einem Ergebnis zu kommen. Das deutlich zu viel Zeit für einen Algorithmus der möglichst 33 mal pro Sekunde die Daten für eine weitere Verarbeitung aufbereitet. Dabei ist in dieser Zeit noch nicht die Zeit zum vor und nachbereiten der Daten miteinbezogen. Clusteringalgorithmen sind poweful, um kleinste Zusammenhänge in Daten zu finden. Da hier die Ähnlichkeit der helleren Daten eigentlich nur ein Mittel zum Zweck ist um den den Elektronenstrahl zu finden, sind Clusteringalgorithmen für diesen Fall zu tiefgründig und zeitintensiv um für den Anwendungsfall.

Ein weiterer Ansatz das Abbild des Elektronenstrahls in den Bilddaten zu finden basiert auf der Verteilung des Elektronenstrahls. Der Kerngedanke dabei ist, das ein Fit-Berechnung mit weniger Dimensionen und Parameter schneller ist als eine komplette Analyse, wie sie am Ende durchgeführt werden soll. Es müssen weniger Möglichkeiten getestet werden um zufriedenstellende Parameter zu finden. Statt also ein Fit für x, y, und z-Werte durchzuführen, werden die Daten vereinfacht betrachtet. Dazu folgenden Überlegungen; Es ist relativ leicht in einem np.array den höchsten Wert zu bestimmen. Der höchste Wert in einer Gauss Glocke ist auch immer das Zentrum der Glocke. Wenn also in dem Bilddatensatz der höchste Wert bestimmt werden kann Zentrum. Median filter sei dank

X-y Achse schneiden, Pixel an dieser stelle 1D analysieren (Name 1D Fit), Width ergibt sich aus sigma und center, kurve finden und zuschneiden.

Verlässlichkeit dieses Verfahrens debatable. Rotation vernachlässigt, höchster Punkt auch debatable

Testung verlässlich? Zeitliche einordnung? Einschätzung

Das 1D Verfahren erfüllt die gewünschte … Weitere Nachforschungen ergaben einen weiteren Ansatz um das Abbild der Elektronen in den Bilddaten zu finden. Er beruht darauf, dass es sich hierbei um Bilddaten handelt, die von ihrer Form als np. Array schon aus, Positionsinformationen der Daten enthalten. Für Bilddaten gibt es von scipy, der Bibliothek von Python für mathematische Funktionen, eine Bibliothek, die sich ndimage nennt. Wird zum Bespiel Analyse von Wolkenbilder verwendet. Ein ein schwarz/weissen Wolkenbild wird danach gefilter wie hell ein Pixel ist. Ist ein Pixel hell genug um eine Wolke zu sein, steht im Ergebnis array eine 1 sonst eine 0. Mit Hilfe der ndimage Funktion label können jetzt alle zusammenhängenden Wolkenpixel bestimmt werden. Dabei müssen keine extra x oder y Werte erzeugt werden, damit der Abstand zwischen den Pixel bestimmt werden kann, wie das beo Clustering der Fall war. Der label Algorithmus kennt die Array Struktur des Bildes und kann auf dessen Basis benacahbarte Pixel bestimmen. Das erfordert weniger Rechenleistung, als jedes Mal eine mathematische Distanz zu bestimmen. Bei dem Wolkenbespiel wäre das Ergebnis der label Funktin ein np. Array der gleichen Form wie die Bilddaten, nur das statt Pixelwerte Nummern für die Wolke zu der sie gehören notiert sind (Abb. ?) Das gleiche könnte man auch auf den Anwendungsfall übertragen. Alle Pixel, die heller als ein bestimmter Schwellwert sind gehören zur Elektronenstrahlabbildung. Auch hier ist es essentiell das vorher der Median Filter die Aureißer eleminiert hat, weil diese sonst das Ergebnis verfälschen würden.

Vorteilig bei diesem Verfahren ist die benötigte Zeit um die Position des Elektronenstrahls zu finden. Unter Zuhilfenahme einer weiteren ndimage Funktion find\_objects, sind die die Bereiche in denen sich eine gelabeltes Objekt finden licht zu bestimmen. Die Verlässlichkeit von diesem Verfahren steht und fällt mit dem Schwellwert. Ein guter Schwellwert liefert ein zuverlässiges Ergebnis. Deshalb sollte dieser einstellbar sein und es die Möglichkeit geben diesen mit einem Algorithmus zu bestimmen. Ein solcher Algorithmus konnte auch gefunden werden. Er ist ein klassischer Lösungsansatz um Vordergrund und Hintergrund voneinander zu trennen. …

Die drei beschriebenen Verfahren, clustering, 1D-Fit und Labeling sind alle drei funktionierende Lösungsansätze um die Position der Elektronenstrahlabbildung zu bestimmen und einzugrenzen. Um noch einmal sicher zu stellen das der zeitliche Vergleich der Verfahren durch unbeabsichtigte Einbindungsunterschiede beieinflusst wurde. Wurden mit dem Python tool timeit noch einmal eine genauere zeitliche Überprüfung durchgeführt. Alle drei Verfahren wurden mit der gleichen roi getestet und dabei mit dem timeit tools überwacht.

Ergebnsi präsi

Schlussfolgerung ziehen.

Eine weiter Überlegung die sich nach verschiedenen Testungen ergab, war wich dicht der Bereich um den Elekronenstrahl gewählt wird. Es soll auch keine Info verloren gehen. Manuell einstellbar um wie weit Area um Strahl erweitert wird. In Abb. Sind noch einmal zusammen fassend die in 2.2 2. Betrachteten Bildbereiche zu sehen. conclusion Abb, was Roi was object, was fit area

Die Fit-Area wird an Fit weitergegeben

2. Fit

- recap Fit Model Parameter etc connection zu 1d Fit

-> Gauss Zuschneiden probebilder, Fit echte Bilder

- x, y werte erstellen, optimieren

- erste Version lmt fit, Rotation missing

- eigener curve fit

- Rotations Test erfolgreich

- wo genau median nochmal erwähnen

- optimieren Fit

- sampling (unter dem Motto vorher schon zuschneiden noch weiter treiben)

- Boundaries

- Startwerte -> init durhclauf anders

- Rechenoperationen stats

Zusammenfassen was für den Nutzer einstellbar ist

insgesamt conclusion Ablauf vlt mit Bild

Klassendiagramm Analyzer

8/9- Seiten