|  |  |  |  |
| --- | --- | --- | --- |
| Module | Parameter | Lokale Variablen | Funktion |
| FindMinMax | Pointlist | Imin, imax, jmin, jmax, iSplits, jSplits, iDistance, jDistance | * Die minimas und maximas der i und j-Werte der Koordinatenwerden gesucht, um den „Rahmen“ des Gitters um das Schachbrett festzulegen * In den ConstantArrays JSplits und ISplits werden die Zellen des Gitters gespeichert. Diese werden über die Distanz der jeweilien Minimalwerte und Maximalwerte geteilt durch die gewünschte Anzahl an Zellen geteilt. |
| SortPointList | iSplits, jSplits, Pointlist | pi,pj | * Die Eckpunkte werden zunächst der Größe nach nach ihren i-Werten Sortiert. Die sortierte Liste wird dann durchgezählt, so dass jeder Punkt seinen Indexwert in I-Richtung bekommt * Danach werden die Eckpunkte der Größe nach nach ihren J-Werten sortiert und bekommen hier ebenfalls einen Index zugeordnet * (Diese Sortierung ist nach jetzigem Stand des Algorithmus vllt nicht mehr zwingend notwendig) |
| GoThroughConvexHulls | iSplits, jSplits, pj | ConvexHull | * Nun wird herausgefiltert, welcher Punkt in welche Zelle des erstelllten Gitters gehört, somit wird eine grobe Vorsortierung der Punkte für den weiteren Verlauf vorgenommen. * In einer For-Schleife welche alle iSplits durchzählt wird die Funktion FindPointsInConvexHull bei jedem Durchgang aufgerufen welche eine Liste mit Associations in die Liste ConvexHull hinzufügt. * Der Funktion werden die momentanen iSplits der Durchzählung übergeben und alle Jsplits. Des Weiteren wird die nach J sortierte Punkteliste übergeben |
| FindPointsInConvexHull | iSplits[[1,ii]], iSplits[[1,ii+1]], jSplits, pj | ConvexHullCell={}, ConvexHullList,ConvexHullCellKeys = <||> | * Eine Liste namens ConvexHullCell und eine Association nachmes ConvexHullKeys wird angelegt * Zwei For-Schleifen werden gestartet. Die erste läuft durch alle Jsplits, die zweite geht alle Punkte von pj durch. * Innerhalb der For-Schleife wird dann überprüft, welche Punkte aus pj sich innerhalb der übergebenen Isplits und den dazugehörigen jSplitts befinden. * Die Koordinaten, die Indizes und die Zellenbezeichnung werden dann in Key in die Association ConvexHullCellKeys gespeichert und er Liste ConvexHullCell angehängt. Diese Liste wird dann and die Liste ConvexHull angehängt   Wiederholung des Vorganges mit neuen iSplits. |
| FindStartVectors | ConvexHull | StartPointCloud={}, StartPointCloudKeys=<||>, VecI,VecJ,countI,countJ,nectI,nextJ, | * Die Zellen (1,1), (1,2) und (2,1) warden in eine neue Liste namens StartPointCloud gespeichert. * Die Liste wird durchlaufen, jedoch nur die Punkte deren Zellenkeys (1,1) aufweisen und der geringste Wert von i und j aus ihnen ermittelt. * Sind beide Werte dem selben Punkt zugeordnet so ist dieser Punkt der StartPunkt für die Gittererzeugung des Schachbretts. (Kissen vezeichnungen, Rotationen) * Bei Tonnenverzeichneten nicht rotierten Bildern, kann es sein das die minimalwerte nicht einem Punkt entsprechen, dann wird der geringste j mit dem nächst geringstem i genommen. * Ist der StartPunkt ermitteln werden die nächsten Punkte in J- und I-Richtung gesucht. * Hierzu wird eine For-Schleife durchlaufen, welche den nächst kleineren Punkt in I- und J-Richtung vom Start aus hat. * Da das noch nicht sicher stellt, das das auch der gewollte Punkt ist, wird noch eine Prüfung und gegebenenfalls Austausch des potentiellen Punktes durchgeführt. * Es wird überprüft der nächste Potentielle Punkt in J-Richtung dahingehend überprüft, ob die Distanz zwischen Start dem neuen Punkt in I-Richtung die minimalste ist, ist dies nicht der Fall werden die anderen Punkte dahingehend untersucht und der Punkt mit dem geringsten Wert genommen. * Selbiges wird auch für die J-Richtung vollzogen * Somit kann man die Richtungsvektoren in beide Richtungen mit den neuen Punkte ermitteln. |
| CreateGrid | nextI, nextJ, Start, ConvexHull | IList={}, JList={}, IDir, JDir, distance, cache, PotNextI, PotNextJ | * IDir und JDir sind die Richtungsvekoren vom Startpunkt aus in beide Kantenrichtungen des Schachbretts. * Danach werden die ersten beiden Spalten in I- und J-Richtung jeweils durchlaufen, und in iList und Jlist gespeichert. * Diese Listen enthalten alle Potentiellen Punkte entlang der gesuchten Kante. * ... |