Der implementierte Algorithmus: Harris Corner Detector

Der Algorithmus:

- void changeSigma(int sigmaSliderValue, void *)
  - void calcWindowWidth() -berechnet windowWidth aus Sigma des Sliders
  - void calcGaussMask()
  - void calcDerivatives()
  - void calcTensor()
- void changeAlpha(alpha_value,0);
  - void calcScore(alpha_value);
- void changeThreshold(threshold_value,0);
  - bool higherThanNeighbour(y,x)

Datenstrukturen:

- struct Pixel – hält berechnete Werte für Pixel
- vector<Pixel> pixValues – hält sämtliche Pixel
- cv::Mat inputImage, workImage, outputImage - Arbeitsmatrizen
- vector<float> GaussMask – GaussMatrix
- int windowWidth – Breite des betrachteten Bildausschnitts / Fensters
- int sigma_value, threshold_value, alpha_value – globale aus Slider berechnete Werte

Vorteile des Algorithmus

- linear Separierbarkeit bringt Geschwindigkeit
  (wurde hier nicht so implementiert)
- erkennt markante Punkte eines Bildes
  → Punkte mit der höchsten Veränderung im Umfeld
- Unveränderlichkeit gegenüber Rotation
- kann lineare Helligkeitsveränderung handhaben
  → erfasst Corner als ebenso markant durch Umgebung

Nachteile des Algorithmus

- Abhängig von der Skalierung und Auflösung
  → je nach Fenster- und Skalierungsgröße verschiedene Scores
- schwierig festzulegen wie Farbwerte aus farbigen Bildern in den Gradienten einzubringen sind (eigene Beobachtung)
  → Maximalwert, Addition, Durchschnitt der Farbvektoren möglich