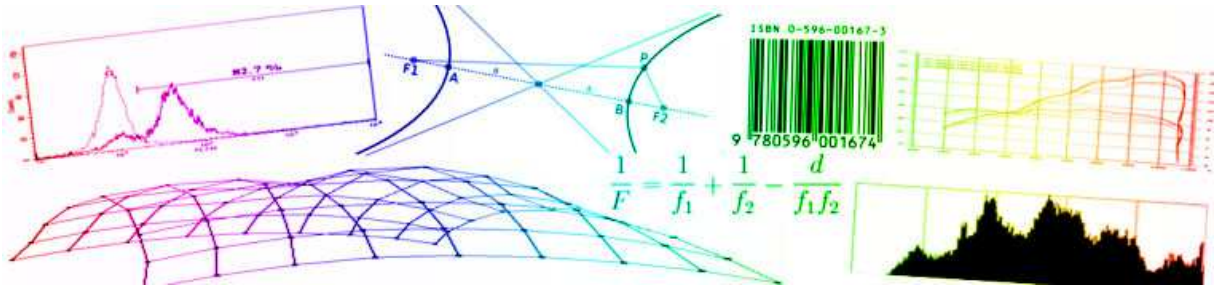
Vision2u



Kostenlose Software zu Bildverarbeitung

Vision2u bietet eine kostenlose Bildverarbeitungs-Software für Privatgebrauch und Forschung. Mit der Software lassen sich Grundaufgaben der Bildverarbeitung bei einfacher Bedienung realisieren. Jeder Webcam-Besitzer kann einfachste Mess-, Zähl- oder Überwachungsaufgaben ohne hohe Investitionskosten erstellen. Derzeit befindet sich die Software in der Beta-Phase. Daher sind einige Funktionen noch nicht vollständig dokumentiert. Das Programm wird ständig um neue Funktionen und Plug-Ins erweitert.

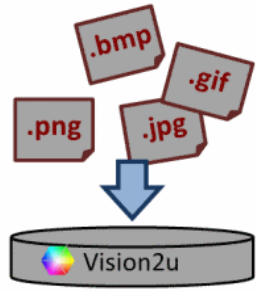
Index

Bildverarbeitung als Dienstleistung

Wir bieten eine automatisierte Analyse von Bildern mit Skripten an. Unsere Experten stellen für Sie einen automatisierten Ablauf mit Vision2u zur Verfügung. Zur Auswertung von Messdaten, Bildanalyse und Vorverarbeitung können Bilder, Fotos und Videos in Farbe oder Schwarz/Weiß verwendet werden. Mit Hilfe der von unseren Experten erstellten Skripte oder Workflows im Batchbetrieb werden die Bildinformationen verarbeitet. Die Tools, mit denen die Bilder automatisch ausgewertet werden, können dem Kunden auf Wunsch zur Verfügung gestellt werden. Auf Kundenwunsch werden die Skripte an neue Anforderungen angepasst oder auf den Systemen des Kunden per Remotezugriff gepflegt und gewartet.

Bildimport

Es können alle gängigen Bildformate (JPEG, GIF, TIFF, BMP, PNG) verarbeitet werden. Dafür stehen Schnittstellen zu Scannern, Mikroskopen, Digitalkameras (TWAIN), Netzwerkkameras, Webcams, Videograbbern, TV-Karten und Videodateien zur Verfügung.

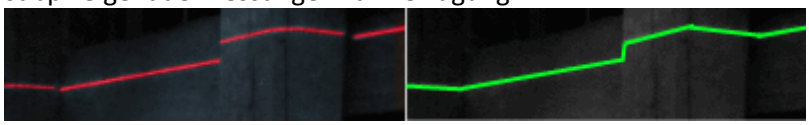


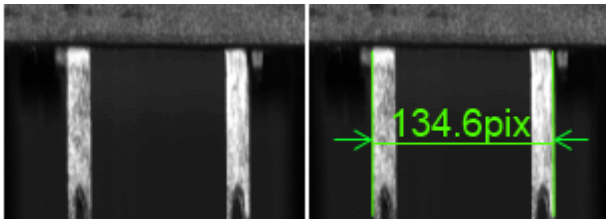
Flächen berechnen

Größen von Objektflächen lassen sich in Pixeln oder kalibrierten Einheiten ausgeben. Es können Flächenverhältnisse und Volumina für Bildstapel oder Videos berechnet werden.

Längen messen

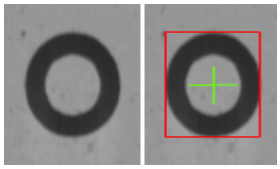
Von Objekten lassen sich Höhe, Breite und Pfadlänge bestimmen. Auch die Länge von gebogenen und verästelten Objekten lässt sich in Pixeln oder kalibrierten Einheiten ermitteln. Für Kanten stehen subpixelgenaue Messungen zur Verfügung.





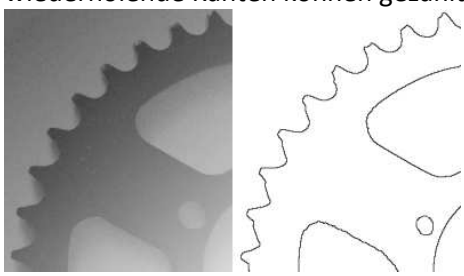
Schwerpunkt ermitteln

Von Objekten lassen sich Schwerpunkt, umschließendes Rechteck, Mittelpunkt und Ausrichtung ermitteln. Die Zahlenwerte lassen sich als Eigenschaften in der Analyse auslesen.



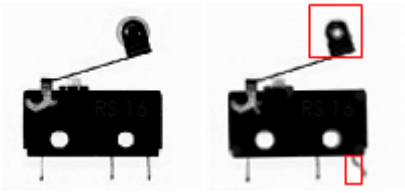
Kanten detektieren

Anhand von Schwellwerten, Kontraständerungen oder Filtern lassen sich Kanten in einer Ebene oder Fläche detektieren. Einige Funktionen zur Kantendetektion arbeiten subpixelgenau.
Von detektierten Kanten lassen sich Abstände, Ausrichtung und Winkel zueinander bestimmen. Sich wiederholende Kanten können gezählt und vermessen werden.



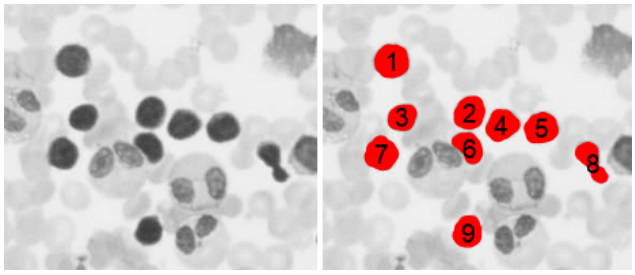
Fehler erkennen

Für die Inspektion im industriellen Umfeld spielt die Fehlererkennung eine wichtige Rolle. Hierfür stehen unterschiedlichste Werkzeuge zur Verfügung. Die Detektion, ob es sich um einen Fehler handelt, kann mittels Farb- /Helligkeitsschwellwerten, Merkmalsextraktion, Formvergleichen (Template Matching), Bildvergleich (Golden Image), Messungen oder Histogrammanalyse ausgewertet werden.



Objekte zählen

In Bildern lassen sich Objekte (Blob) detektieren und zählen. Die Objekte können auf Basis von Masken, Farbfiltern oder Schwellwerten ermittelt werden.
Zusätzlich zur Anzahl der Objekte stehen Informationen zu Länge, Breite, Position, Fläche, Umfang, Form, Ausrichtung und ggf. weitere Eigenschaften zur Verfügung.
Die Daten der Einzelobjekte lassen sich in einer Stapelverarbeitung (Blobanalyse) sortieren und zyklisch auslesen.
Zum Zählen von Kanten und Summieren von Bildinformationen stehen weitere Funktionen zur Verfügung.

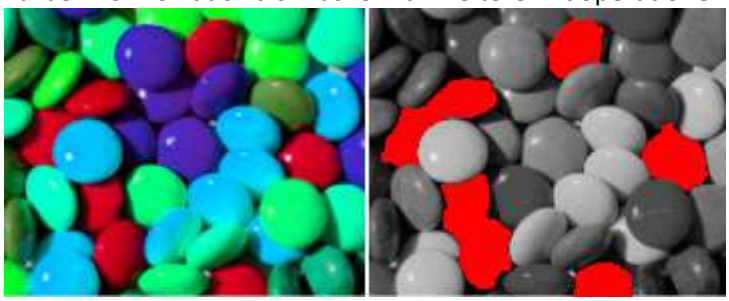


Muster erkennen

Zur Mustererkennung stehen verschiedene Funktionen zur Verfügung. Muster können auf Pixelebene oder auf Basis von Kontraständerungen mit einem Referenzbild (z.B. Passmarken) identifiziert werden. Alternativ können Sie Objekte auch auf Basis ihrer Eigenschaften mittels Featurelisten vergleichen und unterscheiden. Hiermit lassen sich Merkmalsbäume (decision trees) erstellen, die später zur Klassifikation verwendet werden können.
Weiterhin stehen zur Mustererkennung Verfahren zur Schrifterkennung OCR/ORV und zum Lesen von Barcodes zur Verfügung.

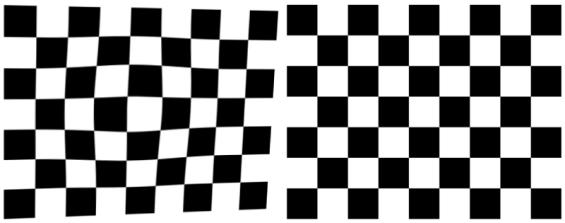
Farben analysieren

Die Farbanalyse filtert und analysiert Farbbereiche. Hierfür können Schwellwerte und Referenzfarben im RGB oder HSL/HSV Farbraum definiert werden.
Farben können auch als Masken für weitere Bildoperationen verwendet werden.



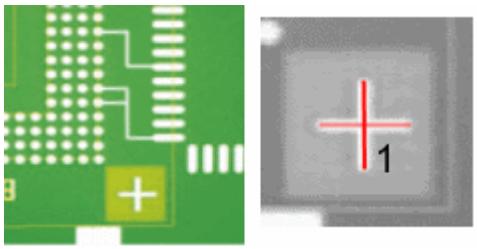
Kalibrierung

Bilder können in realen Einheiten kalibriert werden. Hierzu lassen sich Größenverhältnisse der Pixel in µm, mm, cm, m, km, ae, au, Lj und pc kalibrieren.
Je nach Anwendung ist es möglich, für Längen und Breiten unterschiedliche Kalibrierungen einzustellen.



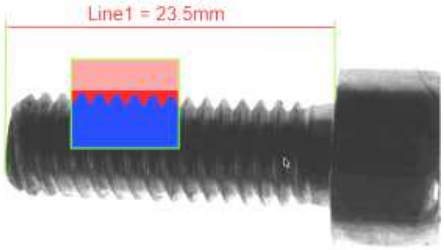
Vorverarbeitung

Bilder lassen sich ausrichten, rotieren und beschneiden.
Es können Helligkeitsverläufe und perspektivische Verzerrungen kompensiert werden. Hierzu zählen die Korrektur von Trapez-, Tonnen-, Kissenverzerrung als auch die Polarkoordinatentransformation.
Bilder können auch vergrößert, verkleinert und zugeschnitten werden.
Bilder können in verschiedene Bildformate konvertiert werden. Sie lassen sich auch in Graustufen, in Falschfarben oder binär umwandeln.



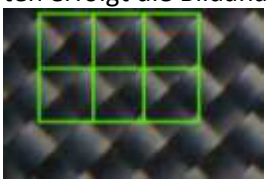
Filter

Mit Filtern lassen sich aus Bildern die relevanten Objekte hervorheben, segmentieren und freistellen. Bildrauschen und Verschmutzungen können zusätzlich mit Filtern reduziert werden. Mit morphologischen Filtern lassen sich Strukturen herausarbeiten und verstärken. Mehrere Bilder lassen sich mit logischen Operatoren verknüpfen.



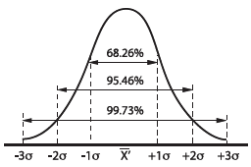
Strukturanalyse

Periodische und stochastische Muster können mit verschiedenen Filtern analysiert werden.
Hierzu können Muster überlagert werden, um die Struktur der periodischen Muster für die Bildverarbeitung aufzubereiten.
Abweichungen von der periodischen Struktur werden in Form von Eigenschaften zurückgegeben. Auf Basis dieser Eigenschaften erfolgt die Bildanalyse.

Qualitätskontrolle

Zur Qualitätskontrolle lassen sich verschiedene statistische Kennwerte ermitteln. Mit Hilfe dieser Kennwerte können Aussagen über die Qualität der Produktion oder Klassifikation getroffen werden. Zur Auswertung lassen sich auch externe Tools mittels CSV oder Excel-Dateien verwenden.



- Genauigkeit (Accuracy)
- Wiederholpräzision/Wiederholbarkeit (Repeatability)
- Vergleichspräzision/Nachvollziehbarkeit (Reproducibility)
- Stabilität (Stability)
- Linearität (Linearity)
- Reinheit (Purity)

Preise

Die Software Vision2u stellen wir Ihnen gerne kostenlos zur Verfügung. Mit Vision2u können Sie Ihren eigenen Bildverarbeitungsworkflow erstellen. Gerne unterstützen wie Sie auch bei ihren eigenen Projekten.
Für die Erstellung eines Script als Dienstleistung kalkulieren wir die folgenden Preise:

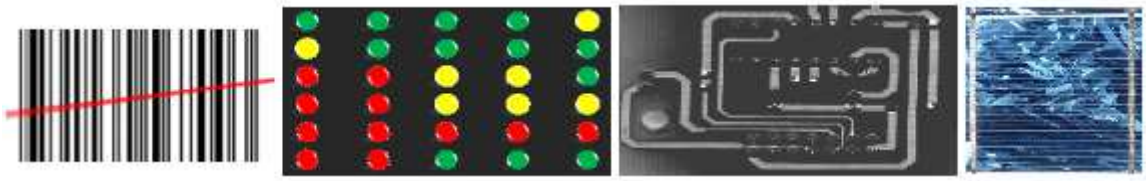
Einzelne Bildverarbeitungsfunktionen:	30€
Schleife oder Verzweigung:	50€

Mit Hilfe des Plugin-Moduls können Sie Vision2u selbst um Algorithmen erweitern.
Aber auch wir bieten die Entwicklung von neuen Filtern oder Bildverarbeitungsfunktionen, passend für Ihre Aufgabe, an.
Fragen sie unverbindlich nach einem Angebot.

Beispiele und Anwendungsgebiete

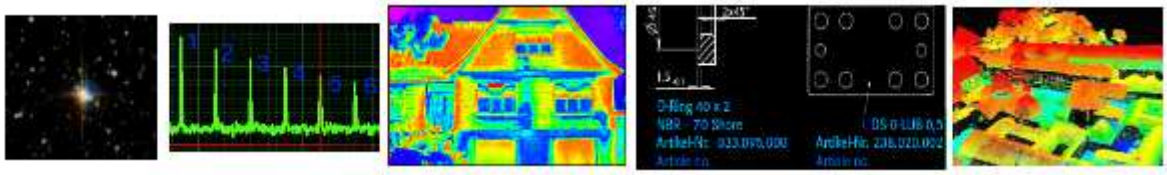
Hier eine Liste von Anwendungsbeispielen für die automatisierte Bildverarbeitung.

Industrie



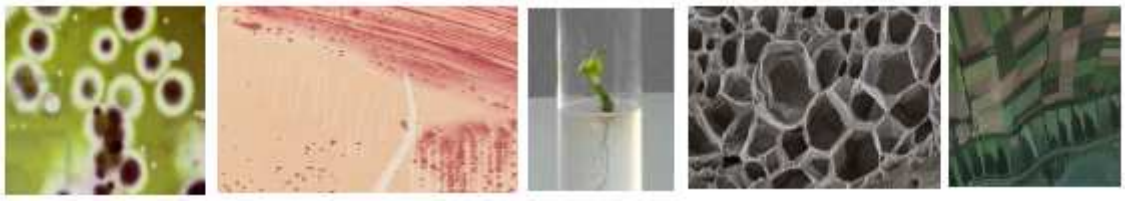
Monitoring und Analyse von Prozessen
Videoüberwachung zur Prozess-Steuerung
Kontrolle von Bauteilen, Zahnrädern, Bohrungen, Lötpunkten und Pins
Qualitätskontrolle von Produkten
Oberflächeninspektion

Forschung und Wissenschaft



Dokumentation und Auswertung von Bildinformationen
Erfassen von Messdaten
Flexibler und schneller Einsatz von Bildverarbeitung für schnell wechselnde Aufgaben
Auswertung von IR-Thermographie-, Röntgenscanner- oder Radarbildern
Messdatenauswertung und Messdatenvisualisierung

Biologie, Medizin, Umwelt

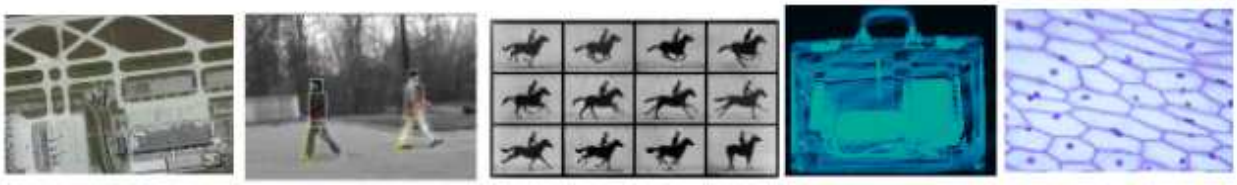


Vermessung von mikroskopischen Objekten
Zählen, Messen und Klassifizierung von Objekten
Partikelform und Partikelgröße
Verteilung von Partikeln, Poren und Mustern
Erfassen und Auswerten von Geodaten
Mikroskopie von Zellen und Gewebe
Analyse von Strukturen

Hobby und Privatanwendungen

Steuerung von Modellen
Überwachung von Eigentum
Automatisierung sich wiederholender Prozesse

Sicherheit



Überwachen von Bereichen und Gegenständen
Auswertung abnormalen Verhaltens
Langzeitüberwachung, Zeitrafferaufnahmen

Arbeitsweise

Um mit Vision2u zu arbeiten, muss zunächst ein Workflow erstellt werden. Dieser Workflow wird mit dem Vision2u-Editor erstellt, parametriert und als XML-File abgespeichert. Mit der Laufzeitumgebung wird dieser Workflow nun abgearbeitet. Als Eingabe stehen Bilder, Videos oder eine Kamera zur Verfügung. Das Laufzeitsystem durchläuft den Workflow nun Schritt für Schritt und nutzt dabei die Bildverarbeitungsbibliotheken, IO-Bibliotheken oder Plugins.

Individuelle Entwicklung

Jedes Projekt ist anders. Sollten für Ihr Bildverarbeitungsprojekt noch wichtige Funktionen fehlen, muss das nicht so bleiben. Nicht alle Funktionen sind auf Geschwindigkeit optimiert. Hierfür ist auch die Performance-Optimierung von Algorithmen möglich. Schon jetzt werden experimentell Funktionen auf der Grafikkarte (GPU) mit CUDA oder OpenCL berechnet.
Die Entwicklung von individuellen Funktionen kann zum Festpreis oder zum festen Stundensatz erfolgen.
Lassen Sie sich bei Vison2u ein Angebot erstellen!

Bachelor- / Masterthesis

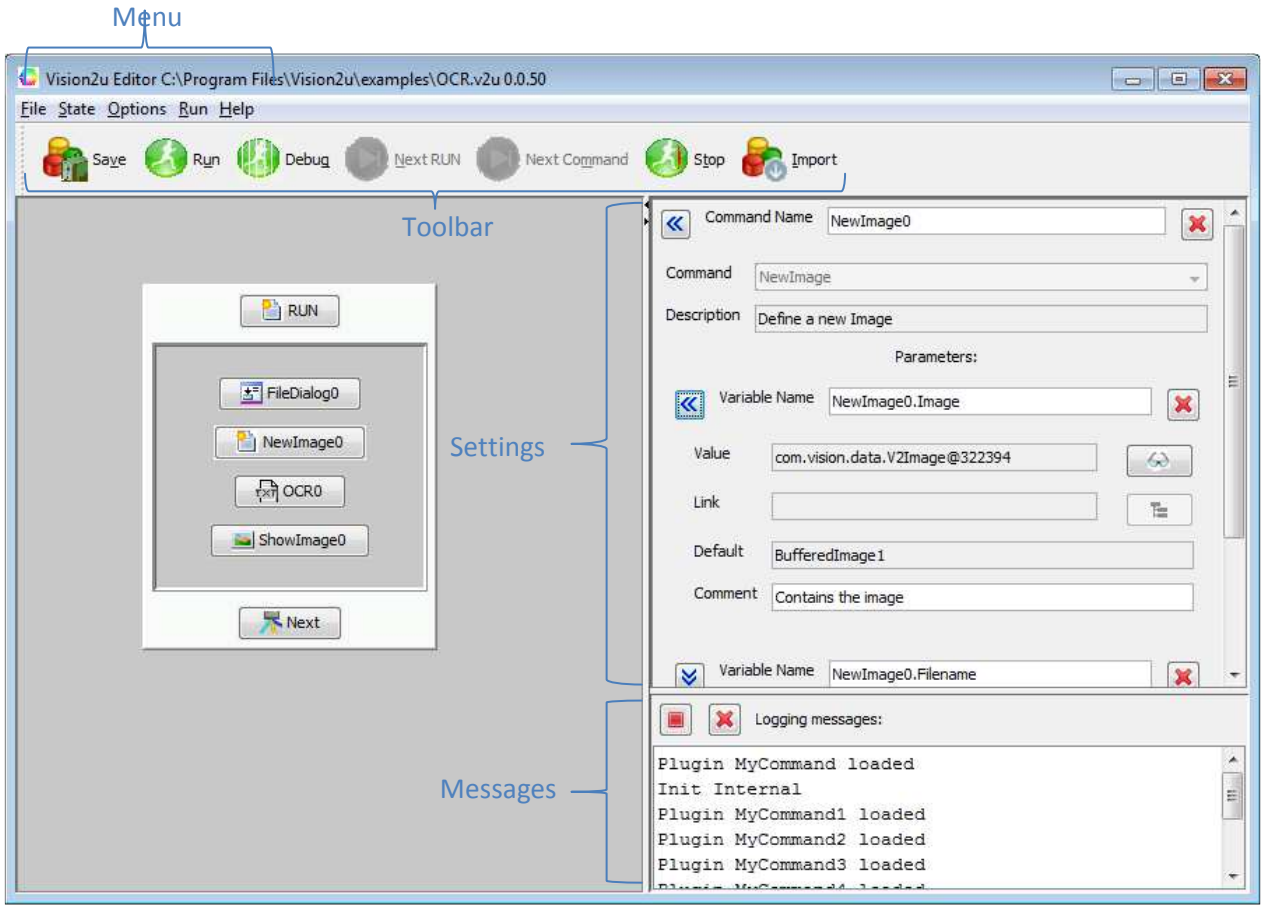
Bei Vison2u entstehen ständig neue Anwendungsgebiete und anspruchsvolle Aufgaben in der Softwareentwicklung. Für Bachelor- und Masterthesis haben wir abgegrenzte Aufgaben zu vergeben. Haben Sie eigene Ideen, die Sie mit Vision2u umsetzen wollen, und benötigen Teile des Quellcodes oder Unterstützung? Dann nehmen Sie Kontakt auf!

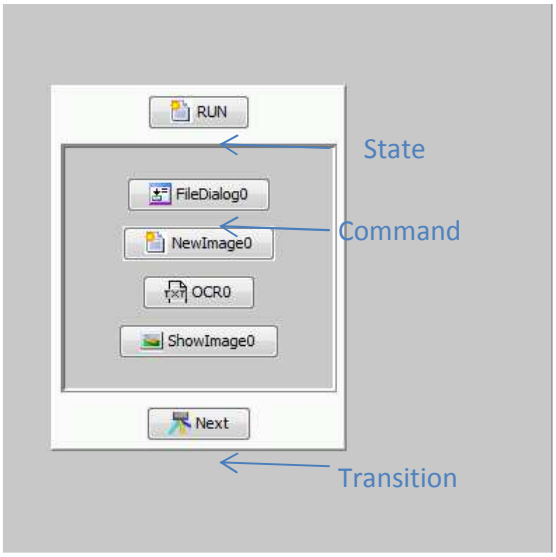
Quick Guide

Ein Bildverarbeitungs-Workflow wird als *.v2u-Projekt gespeichert. Jeder Workflow besteht aus States, innerhalb derer Commands die Bildverarbeitungsfunktionen ausführen.

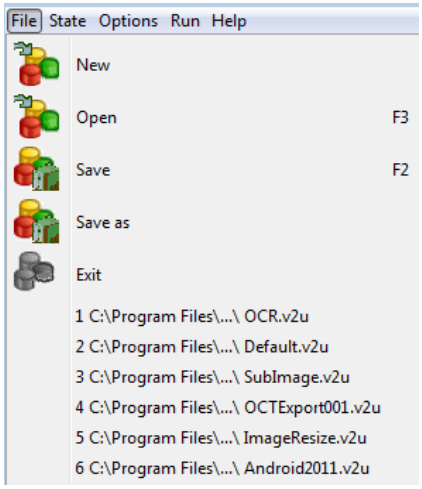
States, Commands, Parameter

Der erste State, der aufgerufen wird, ist der State mit dem Namen ***RUN***. States und Commands haben Parameter, in denen Werte für Eigenschaften verwaltet werden. Einen neuen Command erstellt man über die Import-Funktion. Vor dem Import muss der State ausgewählt werden, in den der neuen Command importiert werden soll. Commands können per Drag-and-drop in der Reihenfolge verändert werden. Parameter können untereinander verknüpft werden, indem in das Feld Link der zu verknüpfende Parameter in eckigen Klammern eingetragen wird. (z.B. [*Edge40.X1*])





Dateimenü (File)



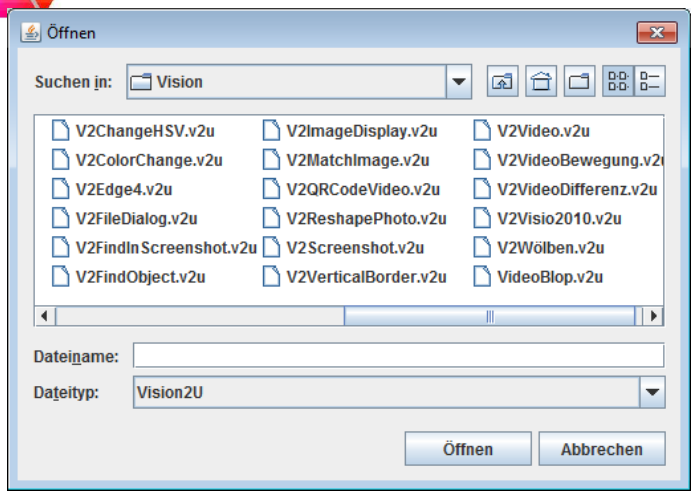
Im Dateimenü lassen sich Vision2u-Projekte laden und speichern und Vision2u lässt sich beenden. Zusätzlich werden die letzten geöffneten Projekte angezeigt. Einige Funktionen lassen sich über Shortcuts erreichen, die hinter dem Befehl stehen.
Zusätzlich steht die Symbolleiste für Befehle zur Verfügung.

Erstellen eines eigenen Projekts (New)

Mit *File->New* lässt sich ein neues Projekt erstellen. Dazu wird ein Template geladen, das eine Grundstruktur mit einem ***RUN***-State bereits beinhaltet.

Öffnen eines bereits bestehenden Projekts (Open, F3)

Vision2u bringt eine Reihe von Beispielprojekten mit. Diese können zum Kennenlernen der Programmfunktionen und zum Verändern genutzt werden.
Bestehende Projekte können über den Menüpunkt *File->Open* geöffnet werden. Die Projekte werden standardmäßig aus dem Verzeichnis Examples aus dem Vision2u-Programmordner geladen.
Vision2u-Projektdateien haben immer die Endung *.v2u*.



Für einige Projekte ist ein installiertes Java-Media-Framework (JMF) oder OpenCV Voraussetzung.

Speichern von Änderungen (Save, F2)

Ein bereits bestehendes oder geöffnetes Projekt kann über *File->Save* erneut gespeichert werden. Dies ist auch über den Shortcut F2 oder über die Symbolleiste Save möglich.

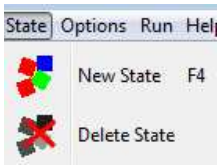
Unter einem neuen Namen speichern (Save as)

Das aktuelle Projekt kann unter einem neuen Namen gespeichert werden, indem man *File->Save as* aufruft. Der Dateiname und Pfad für das Projekt kann dann vom Benutzer angepasst werden.

Vision2u beenden (Exit)

Über *File->Exit* wird Vision2u beendet. Beim Beenden von Vision2u werden Programmeinstellungen, Fensterposition und Fenstergröße gespeichert.

State-Menü (State)



Im State-Menü können neue States eingefügt und auch gelöscht werden. Ein State bietet einen Rahmen, der mehrere Commands beinhalten kann.
Es kann immer nur ein State zu einem Zeitpunkt aktiv sein.
Ein State kann von einem anderen State aufgerufen werden. Der Aufruf zum nächsten State erfolgt über eine Transition (Übergang).
Ein State kann mehrere Transitionen beinhalten. Eine Transition wird nur aufgerufen, wenn ihre Bedingung erfüllt ist. Hat ein State mehrere Transitionen, wird die erste Transition aufgerufen, deren Bedingung erfüllt ist. Alle weiteren Transitionen werden ignoriert.

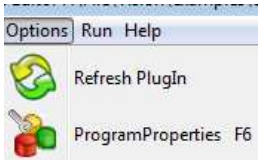
Einfügen eines States (New State, F4)

Über *State->New* State wird im Projaktablauf ein neuer State hinzugefügt.
Der Benutzer wird nach der Eingabe nach einem State-Namen gefragt. Dieser Name darf keinem bereits bestehenden Element entsprechen und keine Sonderzeichen enthalten.

Löschen eines States (Delete State)

Über *State->Delete* State wird ein State gelöscht. Um den richtigen State zu löschen, muss vorher der richtige State mit der Maus ausgewählt werden. Das Löschen erfolgt ohne weitere Rückfragen.
Alle im State befindlichen Commands werden gelöscht.
Wird ein State gelöscht, der in den Projektfluss eingebunden ist, wird möglicherweise das Projekt an dieser Stelle unterbrochen.

Menü Programmoptionen (Options)



In den Optionen können Programmeinstellungen verändert und externe Plugins aktualisiert werden.

Aktualisieren der Plug-Ins (Refresh PlugIn)

Benutzer-Plug-Ins können manuell über das Menü *Options->Refresh* PlugIn aktualisiert werden.
Beim Starten von Vision2u werden alle Plug-Ins automatisch einmal eingelesen.
Für den Fall, dass der Benutzer das Plug-In ändert, während Vision2u läuft, ist das Aktualisieren der Plug-Ins notwendig.
Ein Plug-In wird aber erst aufgerufen, wenn es über Import als neuer Command vorhanden ist.
Eine detaillierte Beschreibung zum Erstellen von Plug-Ins findet sich in der Anleitung unter dem Punkt ***PlugIn***.

Import

Neue Commands lassen sich in einen State importieren.

Diese Funktion ist im Kapitel ***Import*** bei den Programmfunktionen beschrieben.

Programmeigenschaften (ProgramProperties, F6)

Die Einstellungen von Vision2u können über *Options->ProgramProperties* vorgenommen werden. Unter Einstellungen befindet sich die Konfiguration des Programms Vision2u und des aktuell geöffneten Projektes.

Program Properties

Der User Mode dient dazu, verschiedene Benutzerlevel für die Bedienung von Vision2u zu hinterlegen. Im Standardprogramm ist diese Funktion inaktiv.

Mit dem Display Mode kann eingestellt werden, welches Anzeigeprofil der Benutzer erhält.
1 = Alle Eingaben möglich
3 = Anzeige für Entwickler
6 = Anzeige für Benutzer
8 = Eingeschränkte Anzeige

Mit dem Parameter Log Level kann definiert werden, bis zu welchem Detailgrad Ereignisse protokolliert werden.
2 = Nur wichtige Fehler
4 = Fehler und Benutzermeldungen
7 = Alle Anwenderinformationen und Fehler
9 = Wichtige Debug-Informationen einschließlich Anwenderinformationen und Fehlern
11 = Alle Meldungen

Mit dem Parameter Display Level wird festgelegt, bis zu welchem Log-Level protokollierte Ereignisse im Fenster ***Logging messages*** angezeigt werden.

Unter Language lässt sich die Spracheinstellung für die Hilfefunktion setzen.
Die Einstellung ***„en*** entspricht der Sprache Englisch, die Einstellung ***de“***der Sprache Deutsch.

Startup

Unter Startup lässt sich Vision2u entweder mit einem Standardbeispielprojekt (default.v2u) oder mit dem zuletzt geöffneten Projekt starten.

Java Compiler

Um neue Plug-Ins zu kompilieren, muss der Compiler-Path gesetzt werden. Hier muss der installierte Java-Compiler (javac.exe) ausgewählt werden.

Start-Menü (Run)

Im Menü Start lässt sich das aktuell geöffnete Projekt ausführen.
Ein Projekt kann nicht in meheren Instanzen gestartet werden.
Zum Testen des Projektes stehen zusätzlich Einzelschrittfunktionen zur Verfügung.
Das Ausführen eines Projektes kann alternativ über einen Parameter beim Starten von Vision2u ausgelöst werden.
Wird Vision2u geschlossen, so beendet sich auch ein laufendes Projekt.

Ausführen eines Projektes (Run, F9)

Ein geöffnetes Projekt lässt sich über das Menü *Run, F9* oder die Symbolleiste starten. Dabei wird als erstes der **Run**-State aufgerufen. Alle im Run-State befindlichen Commands werden nacheinander abgearbeitet.
Als Komandozeilenparameter kann ein Projekt eingetragen werden, welches mit Vision2u automatisch geladen wird.
Mit dem Komandozeilenparameter ***RUN*** wird das geladene Projekt automatisch gestartet.
In Abhängigkeit vom Projektablauf läuft ein Projket, beendet sich mit dem letzten Befehl oder läuft endlos.

Starten der Einzelschrittausführung (Debug)

Wenn ein Projekt neu erstellt wurde, kann es sinnvoll sein, das Projekt im Einzelschrittmodus auszuführen. Wird der Einzelschrittmodus gestartet, werden alle Commands nacheinander ausgeführt. Zwischen den Commands muss der Benutzer den nächsten Schritt mit der Einzelschrittausführung starten.
Das Projekt wird also jeweils angehalten. In dieser Phase kann der Benutzer die Projektvariablen prüfen.

Springen in den nächsten Schritt (Next State)

Im Einzelschrittmodus kann die Ausführung eines State-Blocks über *Run->Next State* aufgerufen werden. Dabei werden alle Commands innerhalb des States ausgeführt. Erst am Ende des States muss der Benutzer den nächsten State manuell starten.

Springen in den nächsten Command (Next Command)

Im Einzelschrittmodus kann die Ausführung zwischen zwei Commands über *Run->Next Command* gestartet werden.
Dadurch erhält der Benutzer zwischen jedem Command eine Wartezeit, innerhalb derer er die Korrektheit des Projekts prüfen kann.

Anhalten eines laufenden Projekts (Stop, F10)

Ein laufendes Projekt kann mit *Run->Stop* jederzeit gestoppt werden. Alternativ kann ein Projekt auch über *F10* oder die Symbolleiste gestoppt werden.
Dies ist notwendig für Projekte, die in einer Endlosschleife laufen. Es kann aber auch bei der Fehlersuche im Projekt hilfreich sein.
Wurde das Projekt gestoppt, werden alle Projektmodule gelöscht. Die Variablen bleiben jedoch erhalten.
Speichert man ein gestopptes Projekt, werden die aktuellen Variablenwerte mitgespeichert.

Hilfe-Menü (Help)

Im Hilfe-Menü kann der Benutzer ein Hilfefenster einblenden und Informationen über die aktuelle Programmversion erhalten.

Anzeigen der Hilfe (Help F1)

Über das Menü *Help->Help* lässt sich die Hilfe starten.

Versionsinformationen (About)

Im Menü *Help->About* kann der Benutzer die aktuelle Versionsnummer erfahren.
An der Versionsnummer kann man erkennen, ob es auf der Homepage www.vision2u.de eine neue Version gibt.
Zusätzlich erhält man im About-Fenster Informationen, welche Programmteile in welche Programmversion eingeflossen sind (Release Notes).

Installation

Download

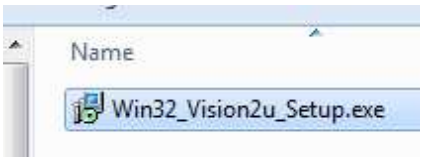
Den Download der aktuellen Version finden Sie unter:
http:/ /vision2u.de/download.htm
Die Software befindet sich noch in der Entwicklungsphase. Während der Entwicklung ist die Software teilweise noch fehlerhaft. Um Vision2u zu Testzwecken und zur Forschung bereits in der Beta-Version auszuprobieren, kann die Software genutzt werden. Es kann jedoch keine Garantie für die Funktionalität der Software übernommen werden.

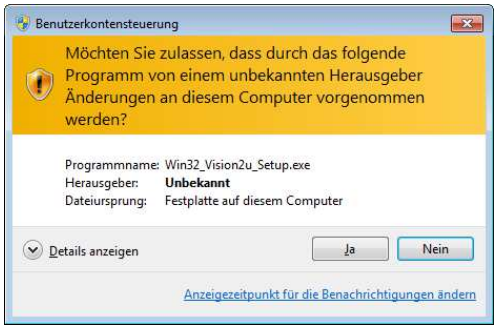
Systemvoraussetzung:

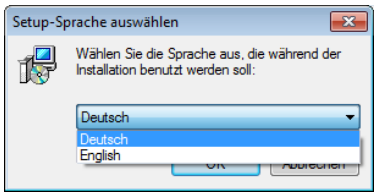
Windows XP, Windows 2000, Windows Vista, Windows 7, Windows 8
Java 7.x – JDK Version 1.7.x

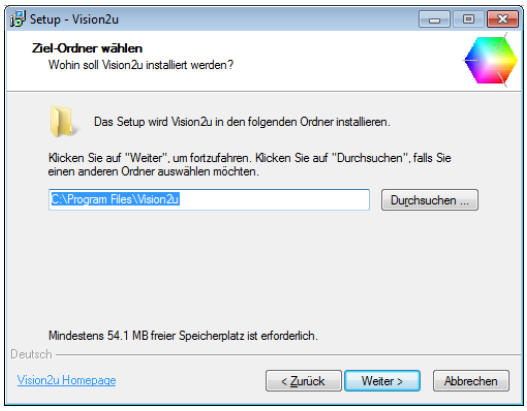
Installation

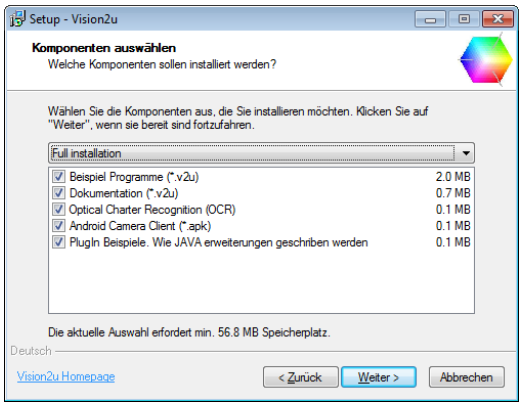
Starten Sie die Setup-Datei wie folgt:

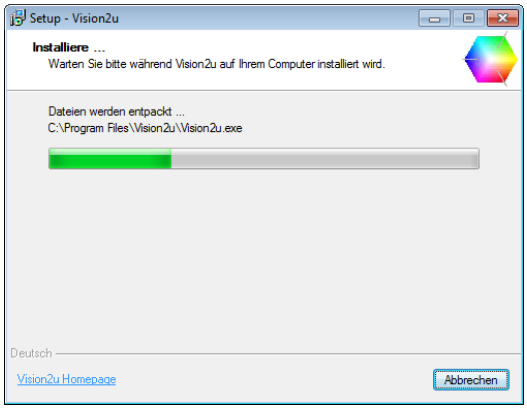


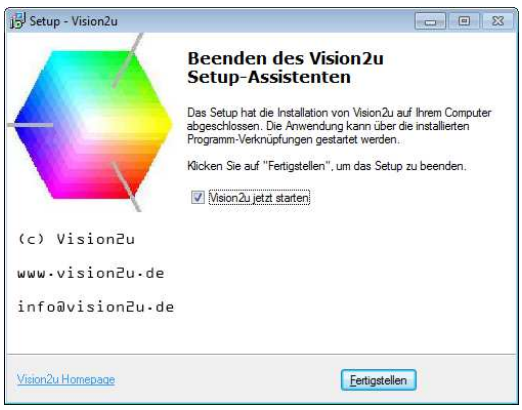








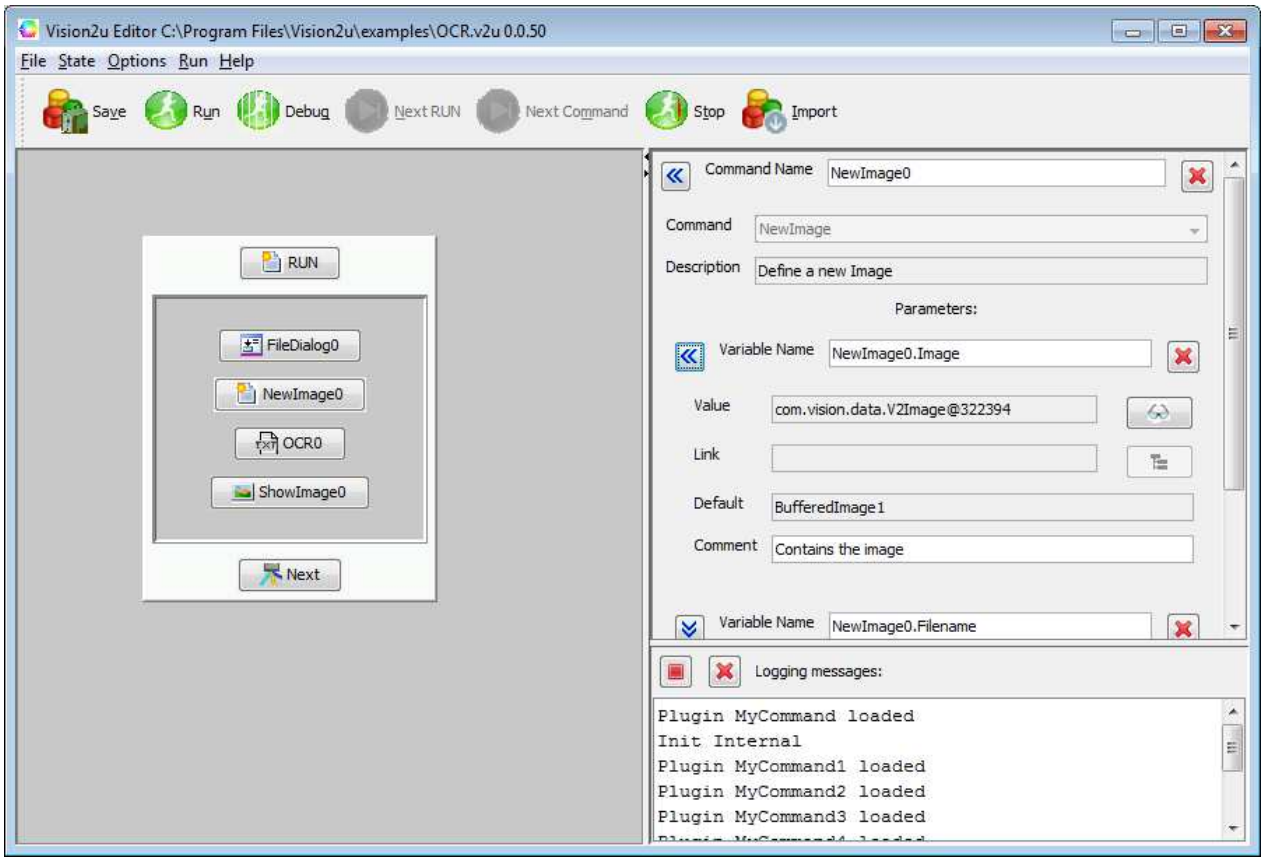




Für die Benutzung einer Webcam ist das Java-Media-Framework API (JMF) zu installieren:
http://java.sun.com/javase/technologies/desktop/media/jmf/

Vision2u kann nach der Installation unter Windows wie folgt gestartet werden:
Start / -->Programme-->Vision2u-->Vision2u

Grundlagen



Idee

Mit Vision2u kann der Benutzer sich einen Bildbearbeitungs-Workflow erstellen. Ein solcher wird als ***v2u***-Projekt gespeichert.
States können untereinander verknüpft sein. Dies kann über Events geschehen, die von Commands ausgelöst werden. Nach Abarbeitung eines States kann auch ein Aufruf über eine Transition 

erfolgen. Werte können innerhalb von Parametern über das Feld Link verknüpft werden. Der zu verknüpfende Wert ist in rechteckigen Klammern [und] einzufassen.

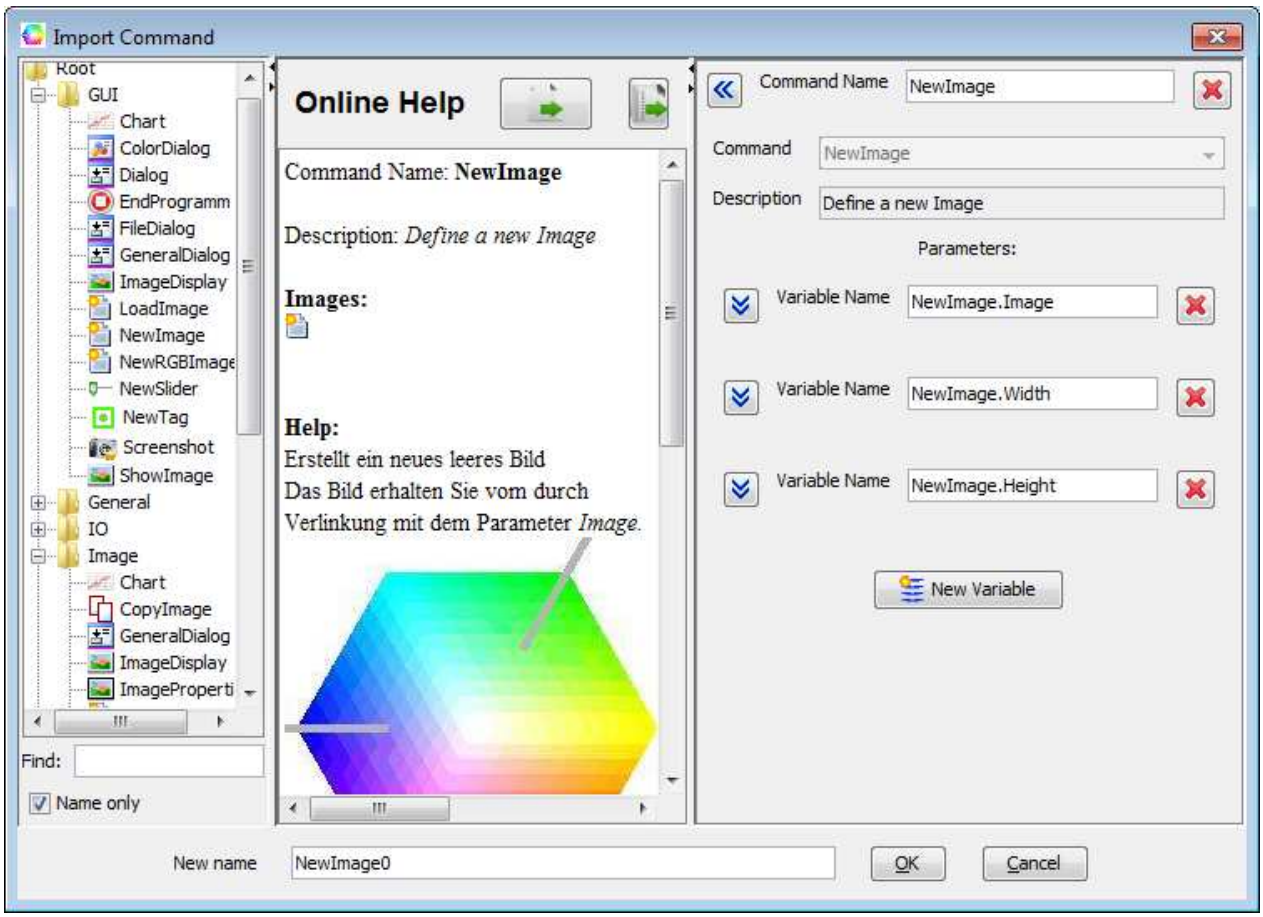
State

Der Bildverarbeitungs-Workflow wird aus States zusammengestellt. Der erste State, der aufgerufen wird, ist der State mit dem Namen ***RUN***. Ein State besteht aus Commands, Parametern und Transitionen. States und Commands haben Parameter, in denen Werte für Eigenschaften verwaltet werden. Über das Menu *State->New* State kann ein neuer State im Workflow hinzugefügt werden. Zu jedem State lässt sich zusätzlich eine Beschreibung (Description) hinterlegen. Diese Beschreibung ist für den Projektablauf ohne Funktion.
Ein Stateblock lässt sich im Workflow per Drag-and-drop frei verschieben. Die Größe eines Stateblocks lässt sich per Drag-and-drop am Rahmen verstellen.
Die Einstellungen eines States werden auf der rechten Seite angezeigt, wenn man auf den Statenamen in dem Stateblock klickt.
Die einzelnen Einstellungen des States lassen sich über Doppelpfeile ein- und ausblenden.
In den Variablen eines States werden dessen Positionen und Größe gespeichert.
Mit dem X-Symbol auf der rechten Seite lassen sich der State, Commands im State, Variablen und Transitionen löschen.

Command

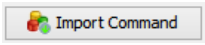
Einen neuen Command erstellt man über die Import-Funktion. Vor dem Import muss der State ausgewählt werden, in den der neuen Command importiert werden soll. Commands können per Drag-and-drop in der Reihenfolge verändert werden. Dazu wird der zu verschiebende Command angeklickt und bei gedrückter Maustaste an die neue Zielposition verschoben.
Parameter können untereinander verknüpft werden, indem in das Feld ***Link*** der zu verknüpfende Parameter in eckigen Klammern eingetragen wird. (z.B. [***Edge40.X1***])

Import



Neue Commands lassen sich in einen State importieren.
Zunächst muss dazu der State ausgewählt werden.

Danach kann der neue Command in den State über den Button 

in der Symbolleiste oder 

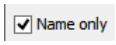
in der Parameterleiste eingefügt werden.

Im Import-Dialog werden auf der linken Seite verschiedene Commands in einer Baumansicht dargestellt. Einige Commands tauchen dabei mehrfach in verschiedenen Zweigen auf.

Nach Auswahl eines Commands besteht im Import-Dialog die Möglichkeit, einen neuen Namen für den Command zu vergeben. 

Mit Hilfe des Textfeldes 

lässt sich nach einem Command suchen.

Ist die Auswahlfläche 

nicht aktiviert, wird der Suchbegriff nicht nur im Command-Namen, sondern auch im Text und allen Parametern gesucht.

In der Mitte des Importdialogs wird die Hilfe zum aktuell ausgewählten Command angezeigt. Über die Schaltfläche  lässt sich die Hilfe zusätzlich in einem Browser öffnen.

Auf der rechten Seite des Importdialogs werden zum ausgewählten Command alle Parameter angezeigt. Der Name des Parameters wird beim Import automatisch mit dem neuen Command-Namen aktualisiert.

Es ist auch möglich, einen bestehenden State als Command anzugeben. An dieser Stelle würde dann der ausgewählte State aufgerufen. Dafür ist im Import-Dialog die Funktion „Show internal Commands“ zu aktivieren.

Run

Ein Projekt kann mit ***Run*** gestartet werden. Mit ***Stop*** lässt sich ein gestartetes Projekt wieder anhalten. Dies ist bei Endlosschleifen notwendig.

Kommandozeile

Vision2u kann auch mit einem Projekt als Aufrufparameter gestartet werden. Die grafische Oberfläche kann dabei deaktiviert werden.

Debug

Zum Testen eines Projektes ist es hilfreich, States oder Commands einzeln nacheinander auszuführen. Um ein Projekt im Debug-Modus zu starten, muss zunächst mit dem Button ***Debug*** begonnen werden. Danach kann die Ausführung State für State mit dem Button ***Next Run*** erfolgen. Alternativ erfolgt die Ausführung command-weise mit dem Button ***Next Command***.

Parameter

Parameter können bei ***States, Commands*** und ***Transition*** eingestellt werden. Mit Parametern wird eine Funktion genauer beschrieben. Die meisten Werte für die Parameter werden bereits beim Import gesetzt.
Für den Projektablauf wird der Wert (Value) eines Parameters verwendet. Es ist auch möglich, Parameter miteinander zu verknüpfen. Hierzu wird im Feld Link der zur referenzierende Parameter in eckigen Klammern eingetragen.

Parameter-Name

Jeder Parameter muss im Projekt einen eindeutigen Namen tragen. Der Name setzt sich zusammen aus Command-Name und Parameterkennung.

Parameter-Value

Ein Parameter hat immer einen Parameterwert (Value). Ist eine Variable vom Typ Input, kann dem Parameter ein Wert zugewiesen werden.
Ist es eine Variable vom Datentyp ***output***, so kann sie nur einen Wert zurückgeben. Der Benutzer kann diesen Wert nicht manuell ändern.
Ein Parameter mit dem Datentyp ***input*** kann Werte gesetzt bekommen und diese Werte als Rückgabewert überschreiben.

Variablen-View

Der Inhalt einer Variablen lässt sich über den Variablen-View anzeigen. Dazu befindet sich bei den Parameterdaten neben dem Parameter Wert (Value) das Symbol  . Wird dies angeklickt, öffnet sich ein neues Fenster, welches den Wertinhalt anzeigt.
In Abhängigkeit von den Datentypen werden die Werte unterschiedlich angezeigt. (Bilder, Texte, Zahlen, Arrays)

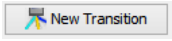
Variablen-Verknüpfung

Ein Parameter kann seine Daten entweder fest aus dem Parameterwert (Value) erhalten, oder sie werden verknüpft. Um eine Verknüpfung herzustellen, muss bei den Parameterdaten ein Link eingetragen werden. Ein Link zu einer anderen Variablen muss in eckigen Klammern stehen.

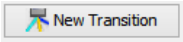
Der Link kann manuell eingegeben oder aus einer Auswahl selektiert werden. Die Auswahl eines Links erfolgt über das Symbol  rechts neben dem Eingabefeld für den Link.
Eine Verknüpfung lässt sich nur auswählen, wenn der Parameter vom Datentyp ***input*** ist.
Die Software prüft nicht automatisch, ob die übermittelten Daten korrekt verarbeitet werden können.
Welche Datentypen geeignet sind, kann im Parameter ***Data Type*** entnommen werden.
Wird kein Link gewünscht, so muss die Variable ***Link*** vollständig leer sein.

Verknüpfungen

Transition

Alle Commands eines Projektes befinden sich innerhalb eines States.
Um innerhalb des Projektes Verknüpfungen, Verzweigungen oder Schleifen herzustellen, können zwei States miteinander verknüpft werden. Diese Verknüpfung nennt man Transition.
Also verbindet eine Transition zwei States miteinander.
Mit dem Button 

kann man innerhalb eines States eine neue Transition erstellen.
Es öffnet sich ein Dialog, um einen eindeutigen Namen für die Transition festzulegen.
Danach öffnet sich der Importdialog, um einen Command als Bedingung für die Transition auszuwählen.
Soll die Transition immer ausgeführt werden, gibt der Command ***Always*** stets einen gültigen Rückgabewert.
Dieser Button befindet sich am Ende der Transition in der Parameterleiste.
Erst nachdem alle Befehle eines States durchlaufen worden sind, werden die Transitionen aufgerufen.
Innerhalb einer Transition existiert genau ein Command.
Ist der Rückgabewert dieses Commands >=1 oder logisch *true*, wird die Transition ausgeführt, d. h. die Transition ruft den im Parameter *Target Name* ausgewählten State auf.
Ist der Rückgabewert des Commands <1 oder logisch *false* so wird diese Transition nicht ausgeführt.
Existieren innerhalb eines States mehrere Transitionen, werden diese nacheinander geprüft. Die erste Transition mit einem gültigen Ergebnis (*true*) wird ausgeführt, indem der State aufgerufen wird. Ist keine der Transitionen eines States gültig, so stoppt das Projekt.
Ein State lässt sich entweder durch Auswahl des 

Buttons intern im State-Fenster oder durch Auswahl 

bei den Settings des States einfügen.

Event

Ein **Event** ist der Aufruf eines anderen States. Dieser Aufruf kann ereignisorientiert erfolgen. So kann beim Eintreten eines Ereignisses (Event) in einen anderen State verzweigt werden. Dies wird z. B. verwendet, um nach einer Wartezeit, dem Eintreten einer Bedingung oder dem Anklicken einer Schaltfläche einen anderen State aufzurufen. Die Verbindung zum Aufruf von einem State zu einem anderen wird mit einer Verbindungslinie angezeigt.

Call

Ein **Call** ist ein Aufruf eines anderen States im Ablauf von Vison2u. Dies funktioniert natürlich nur, wenn im aktuellen Projekt mehrere States existieren. Einen Aufruf eines anderen States kann man einfach importieren. Im Import-Dialog findet man im Import-Baum unter der Überschrift *Internal* alle im Projekt verwendeten Statenamen. Wählt man hier einen State aus, kann man diesen wie ein Command im Projekt platzieren. Kommt das Projekt bei der Ausführung an diese Stelle, wird der definierte State aufgerufen. Die Verbindung zum Aufruf von einem State zu einem anderen wird mit einer Verbindungslinie angezeigt.

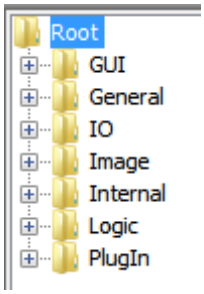
Android-App

Von Vision2u existiert auch eine Android-App. Das Installationsprogramm für die App befindet sich nach der Installation von Vision2u im Programmverzeichnis.
Das Installationsprogramm (Camera Client.apk) muss manuell auf das Android-Gerät übertragen und dort installiert werden.
Die Android-App ist ein Clientprogramm. Auf dem Android-Gerät lässt sich mit einer Kamera ein Bild aufnehmen, das an einen Vision2u-Server übertragen wird. Als Ergebnis kann der Vision2u-Server ein Bild zurück an das Android-Gerät senden.
Dazu muss auf einem Computer in Vision2u das Android-Client-Modul gestartet sein. Dieses befindet sich im Importdialog.
Auf dem Android-Gerät muss zusätzlich der Rechnername oder die IP-Adresse des Android-Servers eingetragen werden.

Funktionen

Neue Funktionen für ein Vision2u-Projekt werden über den Import hinzugefügt. Es stehen verschiedene Kategorien von Funktionen zu Verfügung. Einige Funktionen finden sich in mehreren Kategorien wieder. Aufgrund der Vielzahl von Funktionen sind in dieser Dokumentation nur die wichtigsten Funktionen beschrieben.

Kategorie: GUI
In der GUI-Kategorie finden sich Funktionen, um Bilder anzuzeigen, in Bilder Steuerelemente einzufügen, Diagramme in Form von Charts zu generieren und Benutzerdialoge darzustellen.



Kategorie: General
Always ist eine Dummy-Funktion, die immer einen gültigen Rückgabewert liefert.

Kategorie: IO
In der Kategorie IO befinden sich Funktionen zur Ein- und Ausgabe von Daten in einem Projektablauf.

Computer

Android

Mit dieser Funktion lässt sich eine Verbindung mit einem Android-Endgerät herstellen. Das Android-Gerät sendet das Bild einer Kamera an diese Funktion. In der Variablen [.Image] kann das Bild im Vision2u-Projekt empfangen werden. Die Funktion ruft nach erfolgreichem Empfang den State auf, der in der Variablen [.Event] gespeichert worden ist.
Das Bild kann nun weiter verarbeitet werden. Über die Variable [.Image] wird das Ergebnisbild zurück an das Android-Gerät übertragen. Im Android-Gerät muss für die Kommunikation mit dem Projekt eine Netzwerkverbindung zum Computer bestehen. Zusätzlich muss die IP-Adresse oder der Name des Computers eingetragen werden. Port und Servicename müssen auf Computer und Android-Gerät gleich eingestellt sein.

Beep

Die Funktion **Beep** lässt den Computer kurz piepsen. Hierzu muss in der Windows-Audio-Steuerung dem Hinweiston ein Klang zugeordnet und die Lautstärke aktiviert sein.

EndProgramm

Die Funktion **EndProgramm** beendet das laufende Projekt an einem beliebigen Punkt. Alle nachfolgenden Funktionen oder Verknüpfungen werden ignoriert.

Exit

Mit der Funktion **Exit** wird das laufende Projekt beendet und Vision2u geschlossen.
In der Variablen [.ExitCode] kann ein Rückgabewert für nachfolgende Programme hinterlegt werden.

SaveProject

Beim Aufruf von **SaveProject** werden alle Einstellungen des aktuellen Projektes gespeichert.

Ping

Mit der Funktion **Ping** kann die Erreichbarkeit eines an das Netzwerk angeschlossenen Endgerätes geprüft werden.
In der Variablen [.Host] muss die IP-Adresse oder der Netzwerkname des zu suchenden Endgerätes eingetragen werden.
In der Variablen [.Reachable] wird eine 1 zurückgegeben, wenn das Endgerät erreichbar war, anderenfalls ist der Rückgabewert 0.

 PlayAudio
Die Funktion **PlayAudio** spielt eine Audio-Datei (*.wav) ab. Der Dateiname für das Audiofile muss in der Variablen [.FileName] eingetragen.

 RecordAudio
Mit der Funktion **RecordAudio** lässt sich eine Audiodatei aufnehmen. Der Name der anzulegenden Datei muss in der Variablen [.FileName] definiert werden.
Die Aufnahme kann mit der Funktion **RecordAudioStop** beendet werden.

 RecordAudioStop
Die Funktion beendet eine mit **RecordAudio** gestartete Aufnahme. In der Variablen [.Recorder] muss ein Verweis auf die Recorder-Variable der Funktion **RecordAudio** gesetzt werden.

 SetAudioVolume
Die Funktion **SetAudioVolume** legt die Lautstärke zum Abspielen von Audioquellen fest.
Mit der Variablen [.Volume] wird die Lautstärke eingestellt.

 Execute
Execute führt ein externes Programm aus. In der Variablen [.Command] muss der vollständige Pfad und Programmname angegeben werden. In der Variablen [.Parameter] können zusätzliche Parameter dem zu startenden Programm übergeben werden.
In der Variablen [.Result] wird der Rückgabewert des Programms nach dessen Beendigung abgelegt.
Die Funktion bleibt so lange aktiviert, bis das aufgerufene Programm geschlossen wird.

 FTPDownload
Mit der Funktion **FTPDownload** kann eine Datei von einem FTP-Server heruntergeladen werden. Die Datei wird in das Verzeichnis geschrieben, welches in der Variablen [.Directory] angegeben wurde. Heruntergeladen wird die Datei, die unter der Variablen [.Filename] angegeben wurde. Unter der Variablen [.Server] wird die IP-Adresse oder der Rechnername des FTP Servers angegeben. Mit den Variablen [.User] und [.Password] können die Anmeldedaten gesetzt werden. Bei jedem Aufruf der Funktion FTPDownload wird eine neue Verbindung zum FTP-Server hergestellt.

 FTPUpload
Mit der Funktion **FTPUpload** kann eine Datei auf einen FTP-Server kopiert werden. Die Datei wird in das Verzeichnis geschrieben, welches in der Variablen [.Directory] angegeben wurde. Dieses Verzeichnis muss auf dem FTP-Server bereits existieren. Auf dem FTP-Server müssen die notwendigen Rechte zum Erstellen einer neuen Datei existieren. Unter der Variablen [.Filename] wird angegeben, welche Datei kopiert werden soll. Diese Datei muss mit dem vollen Pfadnamen angegeben werden. Unter der Variablen [.Server] wird die IP-Adresse oder der Rechnername des FTP-Servers angegeben. Mit den Variablen [.User] und [.Password] können die Anmeldedaten gesetzt werden. Bei jedem Aufruf der Funktion FTPUpload wird eine neue Verbindung zum FTP-Server hergestellt.

GetAudioDevices
Mit der Funktion **GetAudioDevices** werden die im Computer vorhandenen Audio-Quellen ermittelt. Für jede gefundene Audio-Quelle wird in der Funktion **GetAudioDevices** automatisch eine neue Variable hinzugefügt. Die ermittelten AudioDevices können in der Funktion **GetAudioLevel** verwendet werden.

GetAudioLevel
Mit der Funktion **GetAudioLevel** wird der Lautstärkepegel eines Audiogerätes zurückgegeben. Die gemessene Lautstärke wird in der Variablen [.AudioLevel] eingetragen. In der Variablen [.Device] muss die Audio-Quelle ausgewählt werden, von der ein Lautstärkepegel gemessen wird. Welche AudioDevices zur Verfügung stehen, kann über die Funktion **AudioDevices** ermittelt werden. Um die Funktion zu nutzen, muss im Windows-Audiomixer eine Audio-Quelle ausgewählt sein.

CatString
Mit **CatString** lassen sich zwei oder mehr Texte (Strings) zu einem Text zusammenführen. Das Ergebnis steht in der Variablen [.Result].
Die Texte, die miteinander verbunden werden sollen, müssen in die Variablen [.S1] und[. S2] geschrieben werden. Sollen weitere Texte angefügt werden, so müssen die Variablen [.S3] usw. bis [.S10] manuell erstellt werden.

GetDate
Mit der Funktion **GetDate** kann das aktuelle Datum ermittelt werden.

GetLPT
Mit der Funktion **GetLPT** können die Eingangspegel an einem Parallelport ermittelt werden. Dazu werden die acht Eingangsbits in die Variable [.Data] geschrieben. Z. B. 00000001=1 00000010=2, 00000111=7.
Welcher Parallelport verwendet wird, kann über die Variable [.Adress] eingestellt werden, z. B.: LPT1, LPT2, 0x378.
Der Parallelport muss zum Einlesen der Daten bidirektional sein. Probleme können bei USB-Parallelport-Adaptern auftreten.

GetTime
GetTime liefert die aktuelle Uhrzeit zurück.

InString
Die Funktion **InString** prüft, ob ein Text innerhalb eines anderen Textes vorhanden ist. In der Variablen [.Result] wird die Position des zu suchenden Textes zurückgegeben. Der zu durchsuchende Text muss in die Variable [.S1] geschrieben werden. In die Variable [.S2] muss der zu findende Text eingetragen werden.

ImageVector
In der Funktion **ImageVector** können mehrere Bilder auf einen Stapel (Stack) gelegt werden. Dazu kann ein neues Bild mit der Variablen [.AddImage] dem Stapel hinzugefügt werden. Der Stapelvektor mit allen Bildern kann mit der Variablen [.Vector] ausgelesen werden.
Mit dieser Funktion werden alle Bilder des Stapels im Arbeitsspeicher gehalten. Dies kann bei vielen Bildern zu Speicherproblemen führen.


Images2Video
Mit der Funktion **Images2Video** kann ein Stapel (Stack) von Bildern in eine Videodatei kopiert werden.


Joystick
Mit der Funktion **Joystick** lässt sich ein angeschlossener Joystick auslesen. Die Variablen [.X], [.Y],[.Z] geben die Achsenpositionen (X, Y, Z) des Joysticks zurück. In der Variablen [.Buttons] werden die Bitwerte aller Tasten am Joystick in ein Byte geschrieben. In der Variable [.POV] wird der Wert der POV Achse (Point of view) zurückgegeben.


Print
Mit der Funktion **Print** wird ein Bild auf dem Drucker ausgegeben.


PrinterDialog
Mit der Funktion **PrinterDialog** lässt sich ein Bild auf dem Drucker ausgeben. Vor dem Druck kann der Benutzer auswählen, in welcher Größe das Bild gedruckt werden soll. Es kann ein Drucker ausgewählt werden. Es kann eine Druckvorschau erstellt werden, oder es kann der Druck abgebrochen werden.


Screenshot
Mit der Funktion **Screenshot** wird ein Bild vom aktuellen Desktop erstellt. Das Bild kann über die Variable [.Image] ausgelesen werden.


Sendkeys
Mit **Sendkeys** können Tastaturbefehle gesendet werden. Es können ganz normale Texte eingegeben werden, die identisch sind mit der Tastaturbedingung.
Die Texte müssen in die Variable [.Keys] geschrieben werden.
Zusätzlich stehen Keycodes für Sondertasten zur Verfügung.

{VK_0} ... {VK_9}	0 ... 9
{VK_A} ... {VK_Z}	A ... Z
{VK_ENTER}	Enter
{VK_SPACE}	Leertaste
{VK_TAB}	Tabulator
{VK_ESCAPE}	Escape
{VK_BACK_SPACE}	Rückschritt
{VK_F1} ... {VK_F12}	Funktionstasten F1 ... F12
{VK_HOME}, {VK_END}	Home, End
{VK_INSERT}	Einfügen
{VK_DELETE}	Entfernen
{VK_PAGE_UP}	Bild aufwärts ↑
{VK_PAGE_DOWN}	Bild abwärts ↓
{VK_DOWN}, {VK_UP}	Cursor aufwärts ↑, Cursor abwärts ↓
{VK_LEFT}, {VK_RIGHT}	Cursor links ←, Cursor rechts →


SetLPT
Mit der Funktion **SetLPT** können Daten auf dem Parallelport ausgegeben werden. Die acht Bits für die einzelnen Leitungen werden in einem Byte in die Variable [.Data] geschrieben, z. B. 00000001=1 00000010=2, 00000111=7.

Welcher Parallelport verwendet wird, kann über die Variable [.Adress] eingestellt werden, z. B.: LPT1, LPT2, 0x378. Probleme können bei USB-Parallelport-Adaptern auftreten.

TCPIPClient
Mit der Funktion **TCPIPClient** können Daten an eine TCP/IP-Schnittstelle gesendet werden. Die zu sendenden Daten müssen in die Variable [.Message] geschrieben werden. Die Verbindung zum TCP/IP-Server wird über die Variablen [.Host] und [.Port] aufgebaut. In die Variable [.Host] ist die IP-Adresse oder der Rechnername des Servers einzutragen. In der Variablen [.Port] wird der Serverport für die Verbindung eingestellt.

TCPIPServer
Mit der Funktion **TCPIPServer** werden Daten von einer TCP/IP-Schnittstelle empfangen.
Die empfangenen Daten werden in die Variable [.Message] geschrieben. In der Variablen [.Port] wird der Port für die Verbindung eingestellt.

Timer
Mit der Funktion **Timer** wird nach dem Ablauf einer Zeit ein State automatisch aufgerufen. Der State wird in der Variablen [.Event] definiert. Das Zeitintervall wird in der Variablen [.Intervall] in Millisekunden eingetragen. Die Funktion wartet nicht auf das Eintreten des Ereignisses.

TimerNow
Mit der Funktion **TimerNow** wird ein Zeitstempel mit dem aktuellen Datum und der Uhrzeit zurückgegeben, z. B. 2012-07-31 08:07:05.604

Video
Die Funktion **Video** nimmt ein Video von einer Kamera auf. In der Variablen [.Image] kann das aktuelle Videobild abgeholt werden. To do testen

VideoFrame
Mit der Funktion **VideoFrame** kann aus einer Videodatei ein Bild geladen werden. Unter der Variablen [.FileName] ist der Dateiname des Videos anzugeben. Mit der Variablen [.Frame] wird definiert, der wievielte Frame als Bild aus dem Video extrahiert werden soll. Das Bild kann über die Variable [.Image] abgeholt werden.

VideoPlayer
Die Funktion **VideoPlayer** spielt ein Video-File ab. In der Variablen [.Image] kann das aktuelle Videobild abgeholt werden.

Wait
Mit der Funktion **Wait** wird eine definierte Zeit gewartet. Die Wartezeit wird in der Variablen [.Time] in Millisekunden angegeben. Der Projektablauf wird für diese Wartezeit unterbrochen.

WebServiceClient
Mit einem WebServiceClient kann man sich mit einem **WebServiceServer** verbinden. Dazu muss auf dem Computer bereits ein **WebServiceServer** gestartet sein. Mit einem verbundenen WebService ist es möglich, Daten zu einem Server zu senden. Hierzu wird die Funktion **WebServiceSetVariable** verwendet.

 WebServiceServer
Ein **WebServiceServer** kann Daten von einem **WebServiceClient** entgegennehmen. Dazu muss der WebServiceClient die Daten mit der Funktion **WebServiceSetVariable** an den Server senden. Der Server muss auf den Empfang der Daten mit der Funktion **WebServiceWaitForVariable** warten.

 WebServiceSetVariable
Wenn ein **WebServiceClient** gestartet worden ist, können mit der Funktion **WebServiceSetVariable** Daten transferiert werden. Zur Verbindung mit dem Server müssen in der Variablen [.Server] und [.Port] der Rechnername/die IP-Adresse und die Port-Nummer des Servers angegeben werden.
Mit der Variablen [.VariableName] wird der Name der an den Server zu übermittelten Variablen angegeben. Dabei wird der Inhalt der Variablen zum Server kopiert.

 WebServiceWaitForVariable
Nachdem ein **WebServiceServer** gestartet worden ist, können Daten mit dieser Funktion empfangen werden. Dazu muss in der Variablen [.Port] ein Port angegeben werden, auf dem der Server auf die Daten warten soll.
In der Variablen [.Variable] werden die Daten der übermittelten Variablen gespeichert.

File

 CopyFile
Mit der Funktion **CopyFile** wird eine Datei kopiert. Dazu wird der Name der zu kopierenden Datei in [.FileNameSource] eingetragen. Mit [.FileNameTarget] wird der Name der zu erstellenden Datei definiert.

 CreateCSV
Mit der Funktion **CreateCSV** wird eine Textdatei erstellt. Hierzu muss mit [.FileName] der Dateiname der zu erstellenden Datei angegeben werden. Der Inhalt für die Datei wird in die Variablen [.Value] geschrieben.
Bei der Variablen [.Append] kann durch die Ziffer 1 definiert werden, ob die Datei neu erstellt oder durch die Ziffer 0 der Text nur angehängt werden soll.
In der Variablen [.Seperator] kann ein Text definiert werden, der zusätzlich zu jedem Eintrag an den einzufügenden Text gehängt wird (Z. B. wird \n zu einem Zeilenumbruch umgewandelt).

 DeleteFile
Mit der Funktion **DeleteFile** kann eine Datei gelöscht werden. Der Name der Datei wird unter [.FileName] angegeben.

 FileDialog
Mit der Funktion **FileDialog** bekommt der Benutzer einen Dateidialog angezeigt. Im Dateidialog kann der Benutzer eine bestehende Datei auswählen oder eine neuen Dateinamen angeben. Dieser Dateiname steht in der Variablen [.FileName]. Werden mehrere Dateien ausgewählt (Mehrfachauswahl), so stehen alle Dateinamen in [.FileNameVector]. Mit der Variablen [.Cursor] kann definiert werden, welcher Dateiname aus dem FileNameVector in die Variable [.FileName] kopiert wird.

Wird die Funktion **FileDialog** mehrfach aufgerufen, so wird der Dateidialog nur beim ersten Aufruf angezeigt.
Sind mehrere Dateien ausgewählt worden, wird der [.Cursor] mit jedem Aufruf um 1 erhöht und der nächste Dateiname in die Variable [.FileName] geschrieben. Dies geschieht, bis alle Dateien durchlaufen worden sind.

 FileExists
Die Funktion **FileExists** prüft, ob eine Datei existiert. Der Dateiname der zu überprüfenden Datei wird in [.FileName] angegeben. In der Variablen [.Exists] steht eine 1, wenn die Datei existiert.

 FileSize
Mit der Funktion **FileSize** wird die Größe einer Datei ausgelesen. Hierzu muss der Dateiname in [.FileName] angegeben werden. In der Variablen [.Size] steht schließlich die Größe der Datei.

 ReadFile
Mit der Funktion **ReadFile** lässt sich ein Text aus einer Datei in eine Variable kopieren.
Die auszulesende Datei muss in der Variablen [.FileName] angegeben werden.
Der Inhalt der Datei steht in der Variablen [.Text] zur Verfügung.

 RenameFile
Mit der Funktion **RenameFile** kann eine Datei umbenannt werden. Der ursprüngliche Dateiname muss in [.FileNameOld] angegeben werden. Der zukünftige Dateiname wird in [.FileNameNew] definiert.

 SaveImage
Die Funktion **SaveImage** speichert ein Bild in Form einer JPG-Datei. Der Dateiname der zu erstellenden Datei wird in [.FileName] angegeben. Das zu speichernde Bild wird in der Variablen[.Image] angegeben.

Image

 Chart
Mit der Funktion **Chart** kann ein Diagramm gezeichnet werden. Dazu müssen die Daten für das Diagramm in Form eines Arrays vorliegen.
Mit der Variablen [.Name] kann für den Chart eine Überschrift definiert werden.

 CopyImage
Die Funktion **CopyImage** erstellt eine Kopie eines Bildes. Das Originalbild wird in [.Image] angegeben. Das kopierte Bild steht in der Variablen [.ImageCopy] zur Verfügung.

 ImageDisplay
Mit der Funktion **ImageDisplay** lässt sich ein Bild anzeigen. Das Bild lässt sich innerhalb der Anzeigevergrößern und verkleinern. In der Anzeige werden Informationen zu dem Pixel des Bilds, auf den der Cursor zeigt, ausgelesen. Der Farbwert innerhalb dieses Pixels, die Cursorposition und die Bildgröße stehen in der Fußzeile der Anzeige.
Es lässt sich weiterhin ein neues Bild laden oder das aktuelle Bild speichern. Im Bild können drei unterschiedliche Markierungstypen vorgenommen werden. Die mit der Markierung selektierten Bildpunkte können aus dem Modul ausgelesen werden. Zur Selektion stehen Punkt, Linie und

Rechteck zur Verfügung. Die Selektion kann auch über die im Modul zur Verfügung stehenden Variablen durchgeführt werden.

ImageProperties

Mit der Funktion **ImageProperties** lassen sich Informationen aus einem Bild auslesen. Das Bild ist in der Variablen [.Image] anzugeben. Vom Bild ausgelesen werden können die Informationen Höhe, Breite, Transparenz und Bildtyp.

Height	Höhe des Bildes
Width	Breite des Bildes
Transparency	Liefert den Typ der Transparenz
Type	Liefert den Farbraum des Bildes (Typ) zurück

LoadImage

Mit der Funktion LoadImage lässt sich ein Bild von der Festplatte laden. Der Dateiname des Bildes ist bei [.FileName] anzugeben. Das Bild steht in der Variablen [.Image] zur Verfügung. Als Dateitypen können *.jpg, *.bmp, *.png, *.gif verwendet werden.

MatchShape

Mit der Funktion MatchShape wird eine Bildform innerhalb eines anderen Bildes gesucht. Die Positionen der gefundenen Bilder werden in den Variablen von MatchShape zurückgegeben. Als Bildform wird die Kontur eines Referenzbildes auf Basis seiner Kontrastwerte gesucht.
To do Funktion testen

MeasureImage

MeasureImage analysiert die Eigenschaften eines Bildes.

Area	Anzahl der Pixel im Bild
HistMax	Liefert den Grauwert der am häufigsten vorkommenden Farbe
HistMin	Liefert den Grauwert der am seltensten vorkommenden Farbe
PixelCount	Anzahl der Pixel im Bild
Max	Farbwert der hellsten Farbe im Bild
Min	Farbwert der dunkelsten Farbe im Bild
Mean	Durchschnittlicher Farbwert des Gesamtbildes
Width	Breite des Bildes
Height	Höhe des Bildes

OCR

Mit der Funktion **OCR** lässt sich ein Text aus einem Bild auslesen.
Die Textfarbe muss dunkel und der Texthintergrund hell sein.
Zum Erkennen des Textes wird die externe Software „Tesseract“ verwendet.
Um neue Zeichensätze einzulernen, müssen die Tools von „Tesseract“ verwendet werden. Diese befinden sich im Verzeichnis \lib\ocr

QRCode

Ein 2D-QRCode kann mit der Funktion **QRCode** entschlüsselt werden.

Dazu muss das Bild mit der Variablen [.Image] gesetzt werden.
Der Text des gefundenen QRCode wird in die Variable [.Result] geschrieben.
Konnte kein QRCode ermittelt werden, bleibt die Variable leer.

ReadBarcode
Aus einem Bild lässt sich mit der Funktion **ReadBarcode** ein Barcode entschlüsseln.
Das zu entschlüsselnde Bild muss in die Variable [.Image] geschrieben werden.
Der Text des detektierten Barcodes wird in die Variable [.Result] geschrieben.
Wenn die Variable [.AutoResize] = 1 ist, wird versucht, das Bild automatisch auf die optimale Größe zur Detektion zu reduzieren.

GenerateBarcode
Das Bild eines Barcodes kann mit der Funktion **GenerateBarcode** generiert werden.
Die Zahl oder der Text, den der Barcode darstellen soll, muss in die Variable [.Barcode] geschrieben werden.
Der Typ des zu verwendenden Barcodes muss in der Variablen [.Typ] festgelegt werden. Zur Auswahl stehen die Typen Code128, Code93, EAN13, Code11, Interleaved2of5, UPCA und UPCE.
Die Variable [.Height] definiert die Höhe des Barcodebildes in Pixeln.
Die Variable [.Width] definiert, wie breit (in Pixeln) ein Barcodestrich werden soll.
Das Bild des generierten Barcodes wird in die Variable [.Image] geschrieben.

QRCodeGenerator
Das Bild eines QRCodes kann mit der Funktion **QRCodeGenerator** generiert werden.
Die Zahl oder der Text, den der QRCode darstellen soll, muss in die Variable [.Text] geschrieben werden.
Die Variable [.Height] definiert die Höhe des Barcodebildes in Pixeln.
Die Variable [.Width] definiert die Breite des Barcodes in Pixeln.
Das Bild des generierten Barcode wird in die Variable [.Image] geschrieben.

NewImage
Die Funktion **NewImage** erstellt ein neues Bild. Mit den Variablen [.Width] und [.Height] werden die Breite und Höhe des Bildes definiert. Das Bild ist durchgängig weiß.

NewRGBImage
Die Funktion **NewRGBImage** erstellt ein neues Bild. Mit den Variablen [.Width] und [.Height] werden die Breite und Höhe des Bildes definiert. Die Farbe des Bildes lässt sich mit den Variablen [.Red], [.Green], [.Blue] für Rot, Grün, Blau definieren.

ShowImage
Mit **ShowImage** lässt sich ein Bild anzeigen. In der Variablen [.Image] muss das Bild übergeben werden.

SwapImage
SwapImage vertauscht zwei Bilder innerhalb zweier Variablen. Die Bilder sind in den Variablen [.Image1] und [.Image2] anzugeben.

StichtImage
Mit der Funktion **StichtImage** lassen sich zwei Bilder zu einem Bild zusammenfügen.
Die Bilder müssen in den Variablen [.Image1] und [.Image2] definiert werden.

Das Bild von [.Image1] wird immer in der linken oberen Ecke des Bildes dargestellt.
Die Position für das zweite Bild wird mit den Variablen [.X], [.Y] festgelegt.
Das Gesamtbild beider Einzelbilder kann in der Variablen [.StichtImage] abgerufen werden.

 FlipImage
Die Funktion **FlipImage** spiegelt ein Bild.
Mit der Variablen [.Mode] = horizontal wird das Bild horizontal gespiegelt. Mit [.Mode] = vertikal wird das Bild vertikal gespiegelt. Das Eingangsbild ist in die Variable [.ImageSource] zu schreiben. Das Ergebnis kann in der Variablen [.ImageTarget] abgeholt werden.

 FloodFillMask
Auf Basis von Schwellwerten lässt sich mit der Funktion **FloodFillMask** aus einem Bild ein Bereich gleicher Helligkeit extrahieren.
In die Variable [.Image] ist das Bild einzutragen.
Mit den Variablen [.X], [.Y] wird der Startpunkt für die Suche des Bildbereichs definiert.
Mit den Variablen [.R], [.G], [.B] können alle darunterliegenden Farbwerte (rot, grün, blau) für die Suche definiert werden.
Beginnend vom Startpunkt werden alle zusammenhängenden Flächen, die dieses Kriterium erfüllen, selektiert.
Das Ergebnisbild wird in die Variable [.Mask] geschrieben.
Das Bild enthält die gefundene zusammenhängende Fläche, und alle anderen Flächen sind schwarz.

 Inverter
Ein Bild wird mit der Funktion **Inverter** invertiert.
Für alle Farbpunkte (rot, grün, blau) wird der Negativwert berechnet (z. B. rot invertiert = 255 – rot).

 ReShape
Ein verzerrtes Bild lässt sich mit der Funktion **ReShape** entzerren. Hierzu wird das Bild in die Variable [.ImageSouce] geschrieben.
Das entzerrte Bild wird in die Variable [.ImageTarget] geschrieben.
Die Größe des Ergebnisbildes wird mit den Variablen [.TargetWidth] und [.TargetHeigth] gesetzt.
Mit den Variablen [.X1] bis [.X4] und [.Y1] bis [.Y4] lassen sich die Eckpunkte des zu entzerrenden Bildes definieren. Das Ergebnisbild beinhaltet nur Bildinformationen, die innerhalb der Eckpunkte liegen.

 SubImage
Die Funktion **SubImage** schneidet einen Bildausschnitt aus einem Bild.
Das zu beschneidende Bild muss in der Variablen [.Image] eingetragen werden. Der Startbereich für den Bildausschnitt wird in den Variablen [.X], [.Y] festgelegt.
Die Breite und Höhe des auszuschneidenden Bildes werden mit den Variablen [.Width] und [.Height] festgelegt.

 CallState
Mit der Funktion **CallState** wird ein anderer State innerhalb des Vision2u-Ablaufs gestartet. Am Ende des aufgerufenen Ablaufs springt die Funktion zurück.

 SetVariable

Über die Funktion **SetVariable** ist es möglich, den Wert einer Variablen zu manipulieren. Der Name der zu verändernden Variablen muss in [.TargetVariableName] eingetragen werden.
Der zuzuweisende Wert muss in der Variablen [.Value] eingetragen werden.

LogicOperator
Die Funktion **LogicOperator** führt Berechnungen mit zwei Zahlen durch. Die Zahlen werden in den Variablen [.1] und [.2] angegeben. Das Ergebnis der Operation steht in der Variablen [.Result]. In der Variablen [.Operator] können verschiedene Operatoren für die Berechnung ausgewählt werden. Zur Verfügung stehen die folgenden Operatoren:

Operator	Funktion	Beschreibung
Bigger	A>B	Wenn A größer B, ist der Ausdruck wahr (true)
Smaller	A<B	Wenn A kleiner B, ist der Ausdruck wahr (true)
Equal	A==B	Wenn A gleich B, ist der Ausdruck wahr (true)
notEqual	A!=B	Wenn A ungleich B, ist der Ausdruck wahr (true)
biggerEqual	A>=B	Wenn A größer oder gleich B, ist der Ausdruck wahr (true)
smallerEqual	A<=B	Wenn A kleiner oder gleich B, ist der Ausdruck wahr (true)
*	A*B	Multipliziert A mit B
+	A+B	Addiert A und B
-	A-B	Subtrahiert B von A
/	A/B	Dividiert A durch B
%	A%B	Berechnet den ganzzahligen Rest der Division A durch B
++	A++	Erhöht A um 1 (Inkrement)
--	A—	Vermindert A um 1 (Dekrement)
>>	A>>	Berechnet den ganzzahligen Wert der Divisionen A durch 2
<<	A<<	Multipliziert A mit 2
&	A&B	Wenn A und B wahr sind, ist der Ausdruck wahr (true)
|	AIB	Wenn A oder B wahr ist, ist der Ausdruck wahr (true)
^	A^B	Berechnet den Wert von A hoch B (Potenz)
SIN	SIN (A)	Berechnet den Sinuswert von A
COS	COS (A)	Berechnet den Cosinuswert von A
TAN	TAN (A)	Berechnet den Tangenswert von A
LOG	LOG (A)	Berechnet den Logarithmus von A zur Basis e
SQR	SQR (A)	Berechnet die Wurzel aus A
ABS	ABS (A)	Schneidet die Nachkommastellen von A ab
INV	INV (A)	Multipliziert den Wert A mit -1
CEIL	CEIL (A)	Schneidet den Ganzzahlwert von A ab
LOG10	LOG (A)	Berechnet den Logarithmus zur Basis 10
MIN	MIN (A,B)	Gibt den jeweils kleineren Wert zurück
MAX	MAX (A,B)	Gibt den jeweils größeren Wert zurück
POW	POW (A,B)	Berechnet den Wert A hoch B (Potenz)
DIFF	DIFF (A,B)	Gibt die Differenz zwischen A und B an

Particles
Die Funktion **Particles** kann vom Untergrund farblich getrennte Objekte finden.
Mit dem Parameter [.MinSize] wird die Größe des kleinsten zu findenden Partikels definiert. Der Parameter [.MaxSize] definiert die maximal erlaubte Größe für einen Partikel.

Zu jedem gefundenen Objekt werden Merkmale berechnet, die in den Variablen (siehe unten) gespeichert werden.
Das Eingangsbild wird in der Variablen [.Image] definiert. Mit der Variablen [.Cursor] kann zum nächsten gefundenen Objekt gesprungen werden. Mit jedem Aufruf der Funktion wird der [.Cursor] automatisch weitergezählt, bis alle Partikel ausgelesen sind.

Operator	Beschreibung
`StartX`	Erste X-Koordinate des Objekts
`StartY`	Erste Y-Koordinate des Objekts
`CenterX`	X-Position der Mitte des Objektes
`CenterY`	Y-Position der Mitte des Objektes
`Area`	Pixelsumme des Objektes
`Height`	Höhe des Objektes
`Width`	Breite des Objektes
`Angle`	Orientierung des Objektes
`Major`	Längste Ausdehnung des Objektes
`Minor`	Kürzeste Ausdehnung des Objektes
`Circ`	Deckungsfaktor zu einem optimalen Kreis
`IntDen`	Maximale Seitenlänge des umschließenden Rechtecks

Random
Die Funktion **`Random`** generiert einen Zufallswert. Der maximale Zufallswert lässt sich mit der Variablen [.MaxValue] einstellen. Die generierte Zufallszahl (zwischen 0 und [.MaxValue]) wird in dem Parameter [.Value] ausgegeben.

Replace
Mit der Funktion **`Replace`** können Zeichen in einem Text (String) ersetzt werden. Der Text wird in der Variablen [.String] definiert. Die zu ersetzenden Zeichen müssen in [.Search] stehen. Der einzufügende Text muss in [.ReplaceWith] stehen.
In der Variablen [.Result] steht der komplett ersetzte Text.

SubString
Die Funktion **`SubString`** extrahiert einen Teilstring aus einem Text. Der Text muss in die Variable [.String] geschrieben werden.
Die Startposition des Teilstrings wird mit der Variablen [.Start] definiert. Mit der Variablen [.Length] wird die Länge des zu ermittelnden Teilstrings definiert. Der gefundene Teilstring kann aus der Variablen [.Result] gelesen werden.


ColorDialog
Der Benutzer kann mit dem **`ColorDialog`** eine Farbe definieren. Die selektierte Farbe steht in den Variablen [.R], [.G], [.B] mit den Werten für Rot, Grün, Blau zur Verfügung.


Always
Die Funktion **`Always`** liefert immer einen gültigen Rückgabewert (true). Dies kann hilfreich sein, um eine Transition, die immer ausgeführt werden soll, zu erstellen. Dazu muss die Funktion **`Always`** als Command in der Transition ausgewählt werden.

FindObject
Die Funktion **FindObject** sucht ein Bild anhand eines Referenzbildes. Die Suche erfolgt auf Basis von Schwarz/Weiß-Kontrasten. Die Suche ist innerhalb gewisser Grenzen rotationstolerant. Auch leichte Größenveränderungen werden noch als Objekt gefunden.
In der Variablen [.Counter] wird die Anzahl der gefundenen Objekte zurückgegeben. Mit den Variablen [.StartX], [.StartY], [.EndX] und [.EndY] werden die umschließenden Eckpunkte des Objektes zurückgegeben.
Über die Variable [.Counter] wird bestimmt, vom wie vielten Objekt die Positionen zurückgegeben werden.

FindImageExact
Mit der Funktion **FindImageExact** wird ein Bild anhand eines Referenzbildes gesucht. Das Referenzbild muss im Bild exakt gefunden werden. In den Variablen [.X] und [.Y] wird die gefundene Position des Objekts zurückgegeben.

CompareImage
Die Funktion **CompareImage** vergleicht zwei Bilder miteinander. Bei zwei unterschiedlich großen Bildern wird nur der überlappende Bereich ausgewertet. Wie stark sich die Bilder unterscheiden, wird in der Variablen [.Differenz] zurückgegeben. Dabei wird der Farbunterschied für jedes Pixel ermittelt.

CompareImageResize
Mit der Funktion **CompareImageResize** werden zwei Bilder in reduzierter Auflösung miteinander verglichen. Mit [.Scale] wird ein Faktor für die Verkleinerung der Bilder eingestellt.
Bei zwei unterschiedlich großen Bildern wird nur der überlappende Bereich ausgewertet. Wie stark sich die Bilder unterscheiden, wird in der Variablen [.Result] zurückgegeben. Dabei wird der Farbunterschied für jedes Pixel in der reduzierten Größe ermittelt.

Resize
Mit der Funktion **Resize** wird die Größe eines Bildes verändert. Mit den Variablen [.Width] und [.Height] wird die Größe des Zielbildes definiert.
Das Ergebnisbild liegt in der Variablen [.TargetImage] vor.

MathSingle
Funktion veraltet. Die Funktion **MathSingle** muss komplett neu geschrieben werden!

String
Mit der Funktion **String** können String-Operationen auf einen Text angewendet werden. Das Ergebnis steht in [.Result].

+	Hängt String1 an String2
Replace	Ersetzt im String 1 den String 2 durch String3
Substring	Ermittelt den Teilstring aus String1 von der Position String2 bis String3
toLowerCase	Wandelt alle Großbuchstaben von String1 in Kleinbuchstaben um
toUpperCase	Wandelt alle Keinbuchstaben von String1 in Großbuchstaben um
Trim	Entfernt alle Leerzeichen aus String1

IndexOff	Gibt die Position von String2 innerhalb des String1 zurück
Equals	Gibt den Wert 1 zurück, wenn String1 den gleichen Text wie String2 enthält

 ToolBar
Mit der Funktion **ToolBar** wird eine Funktionsleiste (**ToolBar**) erstellt.
Mit den Variablen [.X] und [.Y] wird die Position der ToolBar am Bildschirm definiert.

 Button
Mit der Funktion **Button** wird einer ToolBar eine Schaltfläche hinzugefügt. Der Name des Buttons wird mit der Variablen [.Text] festgesetzt.
In die Variable [.Event] ist der Name des States einzutragen, der aufgerufen werden soll, sobald der Button gedrückt wird.

 Lable
Mit der Funktion **Lable** kann ein Text einer **Toolbar** zugefügt werden. Der Text für das Lable wird in der Variablen [.Text] definiert.

 NumberInput
Mit der Funktion **NumberInput** kann einer **ToolBar** ein Eingabefeld für Zahlen hinzugefügt werden.
Der Wert der Zahl kann über die Variable [.Number] gesetzt und gelesen werden.
In der Variablen [.Event] ist der Name des States einzutragen, der aufgerufen werden soll, sobald die Zahl vom Benutzer geändert worden ist.

 TextArea
Mit der Funktion TextArea kann einer ToolBar ein Eingabefeld für Texte hinzugefügt werden.
Der Text kann in der Variablen [.Text] definiert oder gelesen werden.

 Logging Messages

Unterhalb der Parameteransicht werden zur Laufzeit Logging-Informationen ausgeben. Die Logging-Informationen werden vom Entwickler in einem Command integriert, um zusätzliche Informationen anzuzeigen.
Auch Fehler werden im Logging-Fenster ausgegeben. Der Inhalt des Logging-Fensters kann über das X-Symbol gelöscht werden.
Bei vielen Logging-Informationen kann es sinnvoll sein, das Logging über die Taste  anzuhalten.

Plug-In

Es können für Vision2u auch Plug-Ins erstellt werden. Ein Plug-In ist ein selbst erstelltes Java-Programm, das von Vision2u aufgerufen wird.

Externes Plug-In

Ein externes Plug-In kann von jedem Benutzer mit einem Editor erstellt und mit der installierten Version von Vision2u gestartet werden. Alle externen Plug-Ins müssen im Ordner PlugIn im Vision2u-Ordner abgelegt sein. Es ist möglich, das Kompilieren der Plug-Ins aus dem Menü zu starten. Mit dem Menüeintrag *Options -> Refresh PlugIn* werden die Plug-Ins neu übersetzt. Zum Übersetzen muss das Plugin mit JAVA kompiliert werden. Dazu muss der Pfad für die Datei javac.exe unter *Options -> Properties -> Java Compiler* eigetragen werden. Die Datei javac.exe ist der Compiler, der die geschriebene Java-Datei in eine lauffähige Class-Datei übersetzt. Sie können das Java-Development-Kit (JDK) mit dem Java-Compiler von Oracle herunterladen.

Internes Plug-In

Zur Erstellung eines internen Plug-Ins (Moduls) muss der komplette Vision2u-Quelltext vorliegen. Dazu muss Vision2u in der Entwicklungsumgebung geladen werden. Die Entwicklung von Vision2u wurde mit Eclipse durchgeführt. Ein neu erstelltes Modul muss im Package com.vision.engine in der Klasse ModuleLoader registriert werden. Vom ModuleLoader wird beim Starten von Vision2u die Basis-Initialisierung „`public ClassName(){…}`“ ohne zusätzliche Parameter aufgerufen.

Erstellen eines Plug-Ins

Ein Plug-In muss das Interface PlugIn des Package com.vision.plugin.PlugIn implementieren. Dazu sollte der Import com.vision.plugin.PlugIn existieren. Das bedeutet, dass die Methoden init(), start(…) und stop() in der Klasse vorhanden sein müssen.

start(…)

Die eigentliche Funktionalität für das Plug-In wird in der Methode start(…) implementiert. Diese bekommt den Command, also die Daten für die Ausführung, übergeben. Im Command werden alle Parameter für die Ausführung des Plug-Ins und die möglichen Ergebnisse angehängt. Man erreicht diese Parameter über die Abfrage getVariableValue(…) des Commands. Dabei wird ein Objekt zurückgegeben. Dieses Objekt muss gegebenenfalls zur Weiterverwendung umgewandelt (gecastet) werden.

Z. B. `String message = (String) command.getVariableValue("Message", "", true);`

Hier wird der Text des Parameters *Message* abgefragt.

Z. B. `V2Image vImage1 = (V2Image) command.getVariable("Image",null,true).getValue();`
Hier wird ein Bild abgerufen, das unter dem Parameter *Image* abgelegt wurde.

Es können auch Ergebnisse zurückgegeben werden. Ein verändertes Bild oder ein berechneter Wert wird also als Parameter an den Command angehängt. An anderer Stelle im Programmablauf kann dann wieder darauf zugegriffen werden.

Z. B. **`command.setVariableValue("Message", „Hallo Welt“, true);`**
Hier wird in den Parameter *Message* der Text „Hallo Welt“ geschrieben.

Für jeden Aufruf des Plug-Ins wird die Methode **`start(…)`** aufgerufen. Wenn für das Plug-In weitere Klassen gebraucht werden, müssen diese in der Methode **`start(…)`** initialisiert werden. Damit nicht bei jedem Aufruf eine Unterklasse erneut initialisiert werden muss, kann man in Vision2u die Klasse in einem internen Stack ablegen. Mit **`ObjectContainer.getContainerObject`** holt man sich die initialisierte Klasse oder initialisiert die Klasse beim ersten Aufruf automatisch. Da der Rückgabewert des Stacks ein Objekt ist, muss dieses in die entsprechende Zielklasse umgewandelt (gecastet) werden. Um diese Funktion zu nutzen, muss **`com.vision.engine.ObjectContainer`** importiert werden. Die Initialisierung funktioniert nur fehlerfrei für den Basiskonstruktor der Klasse. Um seine Unterklasse im Stack wiederzufinden, wird typischerweise der Name des Commands verwendet. Zusätzlich muss die Klasse zur zu verwendenden Unterklasse angegeben werden.

Z. B. **`TCPIPClient client = (TCPIPClient) ObjectContainer.getContainerObject(command.getName(), TCPIPClient.class);`**
Erstellt eine Variable client vom Typ TCPIPClient. Die Initialisierung erfolgt nur beim ersten Aufruf.

Die Methode **`start(…)`** hat einen Rückgabewert vom Typ double. Dieser wird verwendet, um den Fehlerstatus oder den Ergebnisstatus einer Auswertung zurückzugeben. Ist das Plugin fehlerfrei ausgeführt worden, sollte der Rückgabewert 1 sein. Ist ein Fehler aufgetreten oder ist ein Messergebnis oder eine Berechnung falsch, so sollte der Rückgabewert 0 sein. Prinzipiell sind auch andere Rückgabewerte als 0 und 1 möglich. Dies kann z. B. genutzt werden, um Fuzzy-Analysen durchzuführen.

init()

Die Methode init() wird beim Starten von Vision2u aufgerufen und initialisiert alle Plug-Ins. Dabei müssen auch alle Informationen zum Plug-In erstellt werden, die für die Bedienung des Plug-Ins später notwendig sind. Ein Plug-In wird in Vision2u als Command aufgerufen. Dazu muss zunächst ein Command erstellt werden. Hierzu müssen die Imports com.vision.data.Command und com.vision.data.Variable existieren.
Beim Erstellen des Commands werden dessen Name, eine Beschreibung, ein Hilfetext, eine zugehörige Gruppe, Videos oder Bilder und ein Tooltip übergeben.

Z. B.

```
Command command = new Command(/*name*/this.getClass().getSimpleName(),
/*description*/"Wait for a message via TCPIP socket",
/*help*/"This server is waiting for a line message on port. //n",
/*group*/"IO,Computer",
/*video*/"vimg\\TCPServer.png;",
```

```
/*tooltip*/"A TCPIP server socket");
```

Der Name des Commands muss der Name der Klasse sein, was mit this.getClass().getSimpleName() abgefragt wird. Die Gruppe beschreibt die Sortierung, in der im Import-Baum das Plug-In später angezeigt werden soll. Die verschiedenen Ebenen werden mit Kommas abgetrennt. Um ein Plug-In an mehreren Punkten im Import-Baum anzuzeigen, werden die Begriffe mit Semikolons abgetrennt.

Z. B. „Image,Filter;GUI“
Mit diesem Eintrag in *Group* wird das Plug-In im Zweig *Image* und dessen Unterzweig *Filter* angezeigt. Zusätzlich erscheint das Plug-In im Zweig *GUI*.

Die Parameter, die bei einem Command verwendet werden, müssen zu Beginn abgelegt werden. Die Parameter werden in der Methode **`init()`** dem Command hinzugefügt.
Dies geschieht über die Methode **`command.addParameter(…).`** Hier muss eine neue Variable angelegt werden, die dem Command angehängt wird. Ein neuer Parameter besteht aus Namen, Wert, Kommentar, Beschreibung, Datentyp, Richtung, Tooltip und Hilfetext,

z. B.

```
command.addParameter(new Variable(this.getClass().getSimpleName()+
".Message","Hello World\n","the received message line","","text,string",
"output","",""));
```

Der Name der Variablen entspricht dem Command-Namen, gefolgt von einem Punkt und dem eigentlichen Namen für den Parameter.
Als Wert lässt sich prinzipiell jedes Objekt anfügen.
Der Datentyp wird für eine leichtere Zuordnung von passenden Daten verwendet. Der Text ist nur eine Information, keine automatische Wandlung.
Als Richtung stehen *input* und *output* zur Verfügung. Ein Parameter mit der Richtung *output* speichert Ergebnisse des Plug-Ins und stellt sie in Vision2u zur Verfügung. Hat ein Parameter die Richtung *input,* so kann man dem Plug-In Parameter für die Ausführung übergeben. Es gibt auch Parameter, bei denen es sinnvoll ist, *input,output* als Richtung anzugeben, z. B. wenn ein Parameter einen Wert einliest und verändert zurückschreibt.

Abschließend müssen der Command und seine angehängten Parameter noch an Vision2u übergeben werden. Dies erfolgt mit **`model.addCommand(command);`**.
Hierzu muss der **`Import com.vision.data.ModelLocator;`** existieren.

stop()

In der Methode **`stop()`** können Deregistrierungen erfolgen, die für das Plug-In benötigt werden.

Registrierung des Plug-Ins

Wie das Plug-In aufgerufen wird, wird über den Klassennamen definiert. Wird ein Plug-In als Java-Klasse, z. B. **`TCPIPClient`**, erstellt, steht später die Funktionalität als Command **`TCPIPClient`** zur Verfügung. Um den Command im Import-Dialog anzuzeigen, muss die Klasse noch registriert werden. Dies geschieht in **`com.vision.engine.ModuleLoader`**. Hier werden im Konstruktor der Klasse ModuleLoader alle Plug-Ins registriert.

Z. B.

```
public ModuleLoader(){
                modules = new java.util.Vector<com.vision.plugin.PlugIn>();
                modules.add(new com.vision.tools.ftp.FTPUpload());
                modules.add(new com.vision.tools.ftp.FTPDownload());
                modules.add(new com.vision.tools.TCPIP.TCPIPServer());
                modules.add(new com.vision.tools.TCPIP.TCPIPClient());
}
```

Hier werden die Plug-In-Module **FTPUpload**, **FTPDownload**, **TCPIPServer** und **TCPIPClient** initialisiert.

Weiterentwicklung

Wir versuchen die Software ständig weiterzuentwickeln. In Vision2u können sich aber dennoch kleine Fehler und Bugs eingeschlichen haben. Bitte informieren Sie uns, wenn Sie einen Fehler gefunden haben. Wir werden versuchen, den Fehler baldmöglichst zu beheben. Zum Festpreis oder auf Stundenbasis können wir vielleicht auch Ihre Wünsche in Vision2u integrieren.

Support

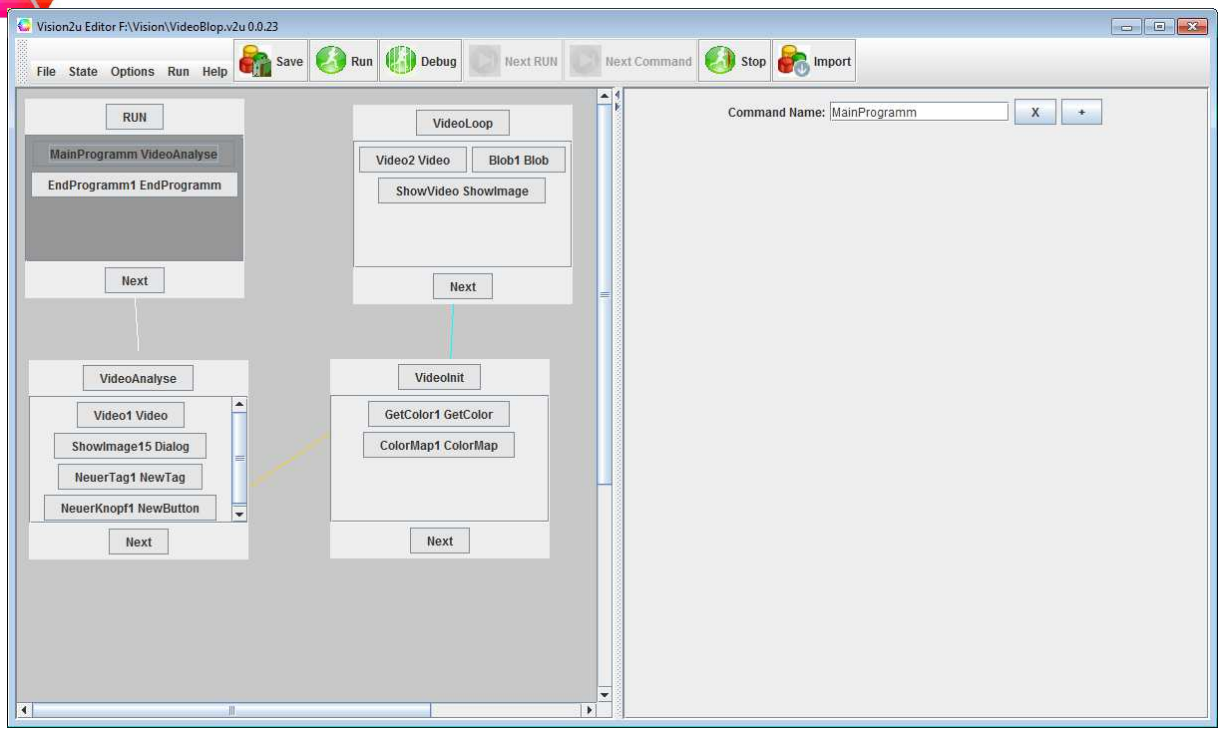
Die Software von Vision2u ist kostenlos. Als nicht kommerzielles Projekt können wir nicht immer alle Supportanfragen sofort beantworten. Dafür bitten wir um Verständnis. Es besteht jedoch die Möglichkeit, Service auf Stundenbasis zu kaufen.

Vision2u
http://vision2u.de

Support Adresse:
Info@vision2u.de
Telefonisch: +49 170 1840859

Patrick Hess
Höflingweg 6
36088 Hünfeld

Steuernummer: 01882730341
U-Steuer-ID: DE-1915092283
Amtsgericht: Fulda



Änderungshistorie

Name	Datum	Änderungen	Version
Patrick Hess	07.10.2010	Entwurf	0.1
Patrick Hess	02.11.2011	Layoutänderung	0.2
Patrick Hess	07.04.2012	Plug-In-Erweiterungen	0.3
Patrick Hess	13.05.2012	Menü Erweiterungen	0.4
Patrick Hess	19.07.2012	Allgemeine Oberflächenbeschreibung	0.5
Patrick Hess	31.07.2012	Funktionsbeschreibung	0.6
Patrick Hess	23.08.2012	Funktionsbeschreibung	0.7
Patrick Hess	11.04.2013	Erweiterung Toolbar, Bildverarbeitung als Dienstleistung	0.8
Patrick Hess	21.04.2013	Funktionsbeschreibung, Bilder eingefügt	0.9
Patrick Hess	24.04.2013	Variablen überarbeitet, Transition neu beschrieben	0.10
Patrick Hess	28.04.2013	Review	0.11
Klaus	08.05.2013	Review	0.12
Patrick Hess	27.05.2013	Erweiterungen Beispiele	0.13