Índice de conteúdos

1. Objectivos

Desenvolvimento de software para implementar um interface do operador de um sistema de visão artificial.

Desenvolvimento de software para aquisição e tratamento de imagem utilizando bibliotecas públicas de funções como o OpenCV.

Desenvolvimento de algoritmos específicos e sequências de operação base para o processamento de imagem em situações industriais típicas.

Implementação de uma plataforma de hardware para realizar o interface com outros sistemas industriais (PLC's, Rede informática da empresa, etc).

2. Motivação

Visão está a tornar-se numa ferramenta importante no panorama industrial. Pode ser utilizada para:

- Controlo de qualidade;
- Identificação de componentes;
- Decisão;
- Reconhecimento de objectos e caracteres.

O conceito de visão por computador data dos anos 30, altura em que uma empresa da industria alimentar concebeu seleccionadores de produtos. Estes eram baseados na utilização de filtros específicos, chamados fotomultiplicadores, e eram usados como detectores.

Hoje em dia é possível verificar muitas características de um produto, graças à utilização de computadores em soluções industriais deste género. Estes, indispensáveis actualmente em qualquer actividade, tornaram-se económicos e fiáveis. Assim, é possível conceber algoritmos complexos, capazes de efectuar grandes quantidades de processamento, com o objectivo de aumentar a qualidade e quantidade de aspectos relevantes a avaliar num produto.

Tal como as suas áreas de aplicação, os métodos de análise de imagens também evoluíram. O famoso DARPA challange – concurso que consiste em efectuar um percurso no deserto por veículos autónomos – é disso um exemplo. O vencedor do ano 2006, da Universidade de Stanford, utiliza como um dos meios de navegação a visão. Também o projecto ATLAS, do Departamento de Engenharia Mecânica da Universidade de Aveiro, utiliza, unicamente, visão para a sua navegação numa pista de obstáculos.

Constata-se que a investigação e desenvolvimento de novos algoritmos e métodos de análise de imagem, demora a chegar às soluções de visão industrial, ficando o seu uso circunscrito aos meios académicos. Desta forma, não é possível evoluir a bom ritmo, neste campo, nas aplicações industriais.

As soluções existentes são de dois tipos:

- métodos analógicos por “sensores”, conectados a câmaras e que são

parametrizáveis. As funcionalidades são em número limitado e as interfaces com o utilizador pouco amigáveis;

- programas que implementam algoritmos de visão. Estes são complexos e com um curva de aprendizagem muito acentuada. A expansão destes programas, a nível de algoritmos de visão, depende do produtor do software e não é possível criar um algoritmo para uma situação específica.

Além da vertente industrial e de desenvolvimento, existe também uma componente didáctica que pode ser aproveitada. Criar uma aplicação de visão no contexto académico, para demonstração e experimentação de algoritmos de visão destinado a alunos, com menores conhecimentos de visão artificial, será uma ferramenta que seguramente promoverá uma linguagem mais amigável e com potencialidades de experimentação.

Pretende-se então criar um software que seja:

- expansível - para integrar novos algoritmos e soluções;
- versátil - para se adaptar a qualquer problema;
- intuitivo – de modo a facilitar a interação com o utilizador;
- comunicativo – para integração no ambiente industrial (PLCs, ethernet)
- rápido – para que nenhum processo fique dependente da acção da aplicação.

3. Metodologia

3.1. Operações e Tarefas

O trabalho desenvolvido baseia-se na definição dos termos Operação e Tarefa.

Para a resolução de problemas de visão, é necessário realizar alguns procedimentos como tais como: filtros na imagem para eliminação de ruído, binarizar a imagem, contar o número de elementos na imagem, etc. Estes, como procedimentos elementares, têm por definição “Operação”.

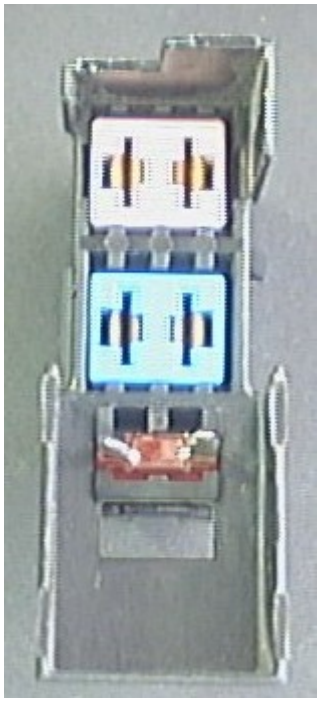

Figura 3.1 – Suporte com 3 fusíveis.

O que se pretende não é tornar a imagem a preto e branco, não é filtrar cores, pretende-se sim, contar o número de objectos existentes, verificar a presença de um objecto ou a sua forma. É necessário relacionar uma sequência de procedimentos para obter o resultado desejado. A um grupo de “Operações” relacionadas dá-se o nome de “Tarefa”.

Em resumo: “Operação” é uma acção elementar, “*Tarefa”* é a associação de várias “Operações” e/ou outras “Tarefas”.

Um exemplo: detecção de 3 fusíveis de cores diferentes num suporte (Figura 3.1).

Criam-se três tarefas: “detectar o fusível azul”, “detectar o fusível vermelho” e “detectar o fusível rosa”. No entanto, desejamos executar as três detecções sempre seguidas num exacto momento. Então, criamos uma nova “Tarefa”, chamada, por exemplo, “Detectar Fusíveis” e adicionamos as três “Tarefas” acima referidas.

Basta agora executar a tarefa “Detectar Fusíveis” que ela se encarrega de executar as “sub-tarefas”.

3.2. Arquitectura do Programa

O programa foi estruturado de forma modular, permitindo a expansibilidade, sendo este um dos requisitos.

São necessários módulos, para incluir mais "Operações", mais interface de

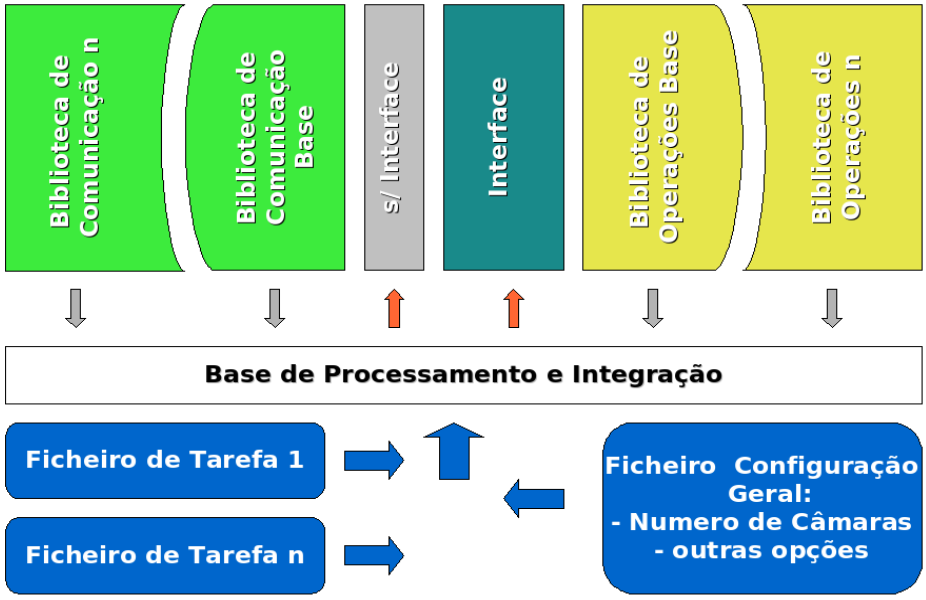

Figura 3.2 – Modelo da estrutura do programa

comunicação e para poder ter vários tipos de interacção com o operador: nenhuma/automático, interface gráfica ou remota). Assim, foi criado o modelo representado na figura 3.2:

De acordo com o modelo, temos dois níveis funcionais e um Auxiliar:

- Base de Processamento e Integração (Base) como nível base;
- Módulos de Expansão como nível superior (Bibliotecas de comunicação, Interfaces, Bibliotecas de Operações);
- Ficheiros de configuração e backup como nível auxiliar (Ficheiros de Tarefas, Número de Câmaras, outras opções).

Toda a manipulação de estruturas de dados encontra-se na "Base", onde é gerido todo o baixo nível ("Operações", "Tarefas", acesso ao Hardware, "Comunicações"). No nível superior da "Base" encontram-se os módulos de expansão:

- Biblioteca de Operações – para adicionar operações à aplicação;

- Biblioteca de Comunicações – para adicionar protocolos de comunicação, comunicação com hardware específico e com outro software;
- Interface gráfica com o operador;
- Sem interface - Módulo para funcionamento sem interface gráfica.

A acrescentar o nível auxiliar, que engloba:

- Gestão de configurações;
- Exportação/Importação para XML – linguagem preferencial para guardar dados em ficheiro;
- Exportação de registos: mensagens de aviso, erros e informações.

Depois de definir o modelo geral, foi necessário definir modelos específicos nomeadamente, relação entre “Operações” e “Tarefas” e para as “Comunicações”.

3.3. Relação entre “Operação” e “Tarefa”

A relação entre Operações e Tarefas é descrita na figura 3.3:

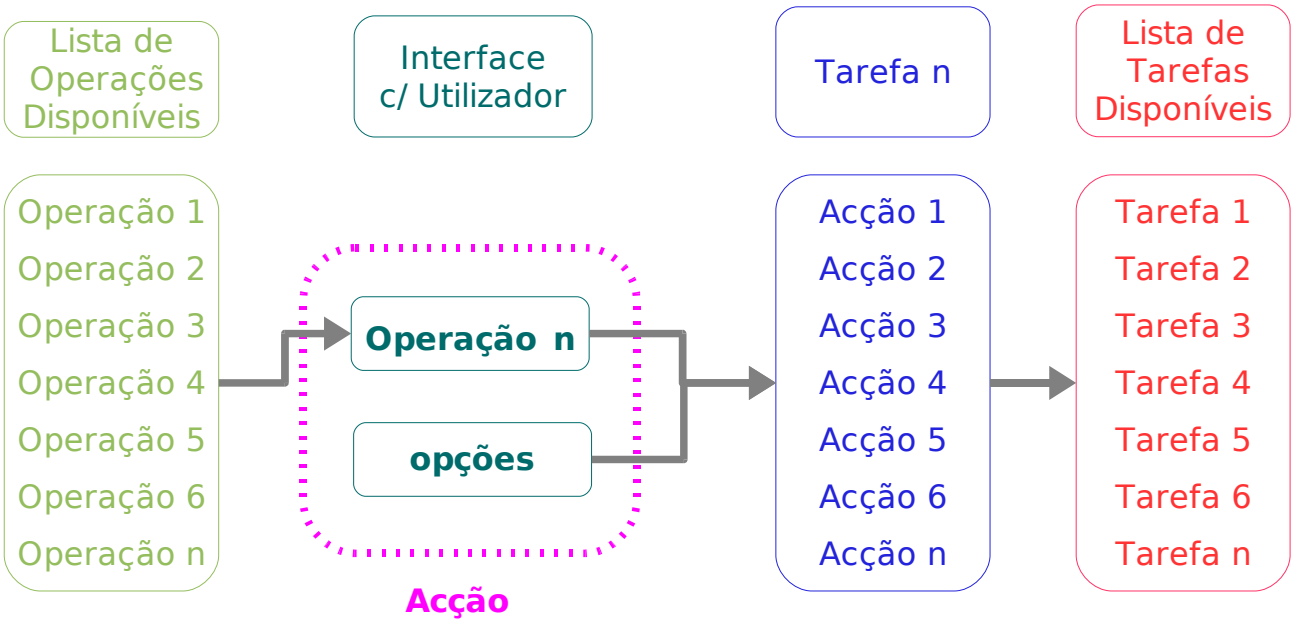

Figura 3.3 – esquema de relações entre Operações, tarefas e opções.

A gestão de Operações e tarefas na aplicação é implementada a três níveis:

- Operações
- Tarefas
- Opções – pois cada Operação precisa ser parametrizada.

Existem duas Listas:

- Lista de Operações Disponíveis
- Lista de Tarefas Disponíveis

A primeira é gerada no início da aplicação com as Operações associadas. É uma lista estática, não sendo actualizada durante a execução da aplicação.

A lista de “Tarefas”, é uma lista onde são adicionadas ou removidas tarefas. Ao criar uma tarefa o utilizador estará a adicionar uma nova tarefa a esta lista.

Para uma “Operação” pertencer a uma “Tarefa” necessita ser parametrizada pelo utilizador. A “Operação” pode apresentar-se de modo que o utilizador necessite de introduzir configurações . Exemplo: o limite numa binarização.

A título exemplificativo é mostrado na figura 3.3 a “Operação” como uma “Acção” que é adicionada à “Tarefa”. Essa “Acção” é o resultado da conjugação de uma “Operação” com “Opções”.

3.4. “Operação”

Foi definido um modelo para a operação: entradas e saídas.

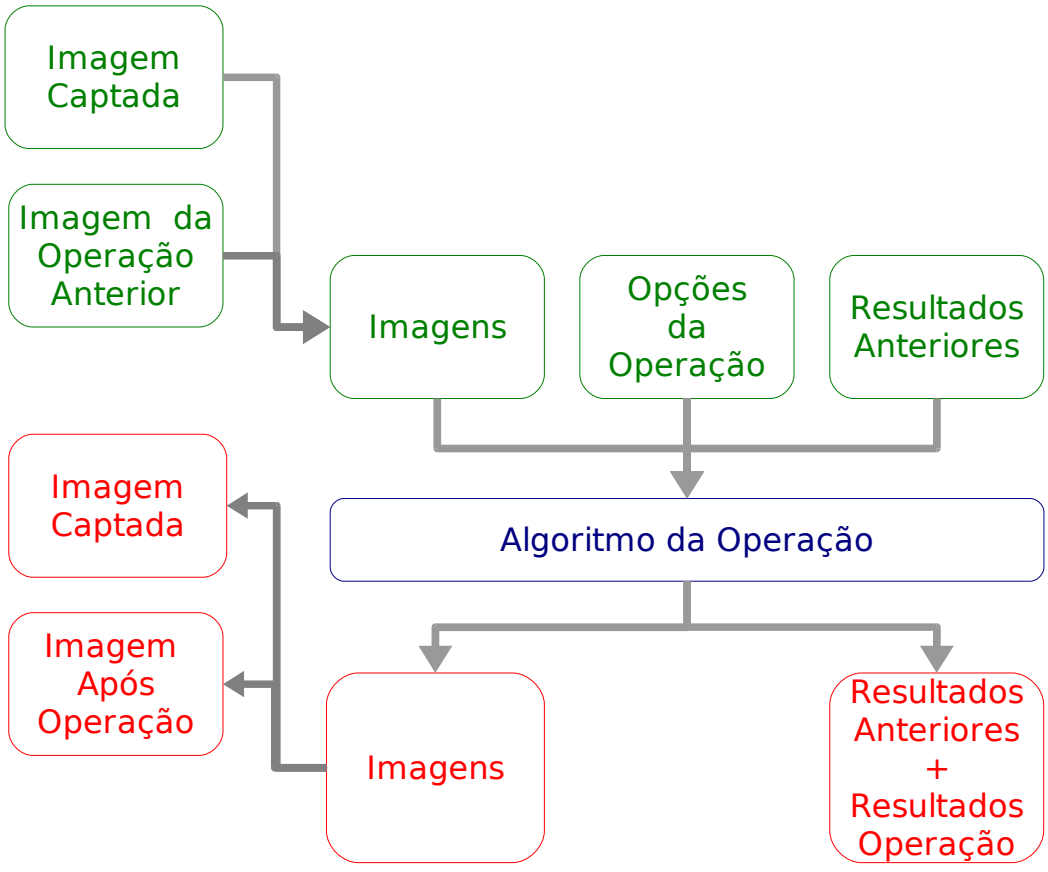

Figura 3.4 – Esquema de um operação

Para o processamento de imagem necessitamos de:

- Uma sequência de imagens, para que a operação possa utilizar a imagem da operação anterior
- Uma sequência de dados, para que a próxima operação possa utilizar dados exportados da anterior (por exemplo pontos X,Y)
- Receber as Opções da Operação.

3.5. "Comunicação"

Outro componente muito importante da aplicação, com planeamento cuidado é a base de comunicações.

A base de comunicações assenta em três níveis (figura 3.5):

- Baixo nível – interacção com bibliotecas externas à aplicação (ex: acesso a módulos ADAM).
- Nível de abstracção – em que uma camada intermédia faz ligação entre a aplicação e o baixo nível.
- Nível Superior – Funções de alto nível que identificam o elemento de comunicação com um único endereço.

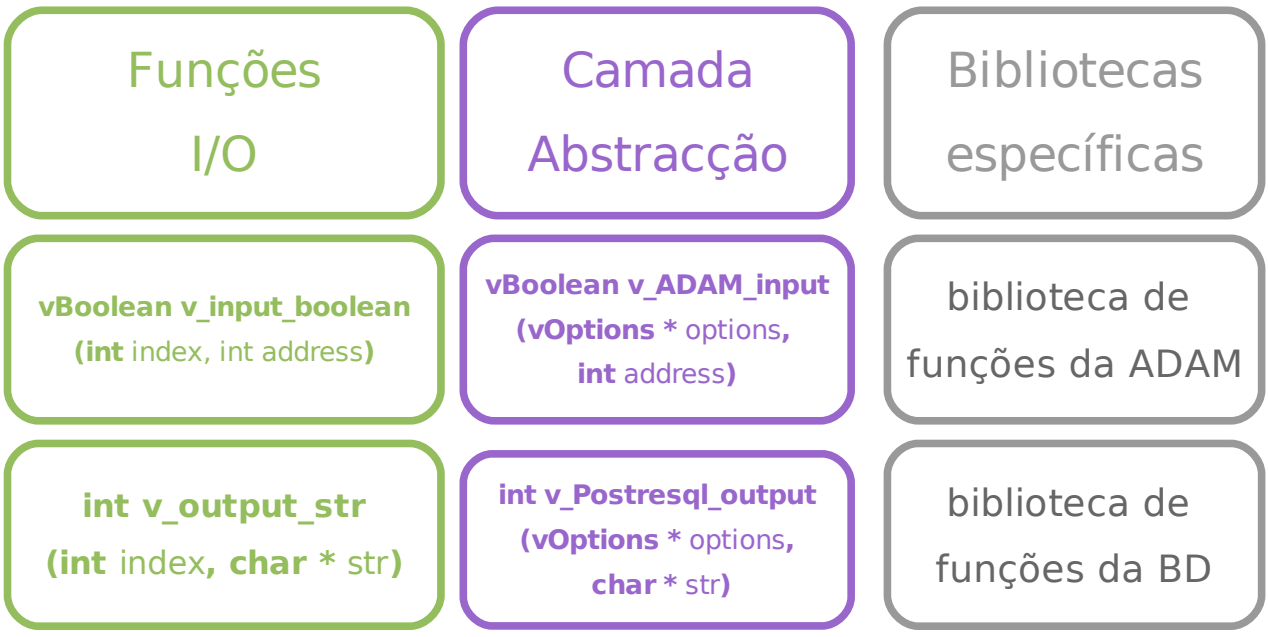

Figura 3.5 – Modelo de funcionamento da camada de comunicações

O objectivo principal deste modelo de funcionamento, é permitir ao programador ou utilizador aceder através de uma função ou operação, a qualquer interface de

comunicações.

O funcionamento das comunicações é equivalente ao das operações e tarefas, existindo duas listas:

- Comunicações suportadas – módulos de comunicações disponíveis na aplicação;
- Comunicações configuradas – comunicações parametrizadas pelo utilizador e disponíveis para serem utilizadas.

A primeira lista, Comunicações Suportadas, é carregada no início da aplicação, com os módulos existentes, e é "estática". O utilizador pode adicionar/remover comunicações configuradas.

As comunicações dizem-se configuradas, quando a elas são associadas as opções necessárias para a comunicação. No exemplo da figura 3.5 é necessário configurar o IP tanto do Módulo da ADAM como do computador, onde a base de dados se encontra em funcionamento.

Na figura 3.6 podemos ver como se relacionam as comunicações (os elementos denominam-se IO's) suportadas e configuradas.

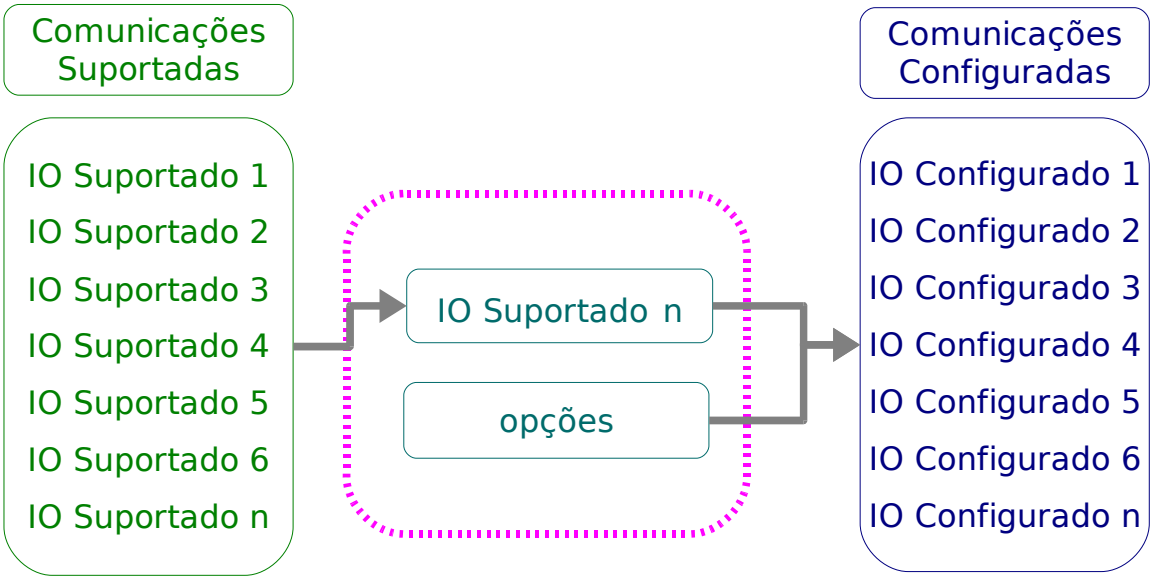

Figura 3.6 – Modelo de Comunicações Suportas / Configuradas

4. Desenvolvimento

4.1. Utilização de bibliotecas Opensource

A aposta desde o início do desenvolvimento foi de aproveitar aquilo que existe disponível sob licenças livres. Existem bibliotecas de funções com elevada qualidade, com licenças deste género. Foram utilizadas essencialmente 4 bibliotecas e 1 aplicação:

- OpenCV – é uma biblioteca de código aberto para Visão por Computador, originalmente desenvolvida pela intel. Está disponível sobre uma licença BSD (Berkley (. Tem diversas funcionalidades para pré e pós processamento de imagem. Tem também funcionalidades avançadas como *Motion Tracking* e detecção de faces. É desenvolvido em C.
- Gtk+ - *gimp toolkit* - Toolkit gráfico originalmente desenvolvido para substituir o Motif (ambiente gráfico utilizado em ambientes unix). Encontra-se disponível sobre uma licença LGPL (GNU Lesser General Public License) e engloba também sub-bibliotecas, nomeadamente a *glib* que é responsável por ferramentas de baixo nível nomeadamente para criar funções mais seguras em relação àquelas disponibilizadas pelo ANSI C e também para gestão de erros ou criação de listas dinâmicas.
- libglade – é uma biblioteca que permite utilizar ficheiros XML para definir interfaces Gtk+. Desta forma, utilizando um editor de interfaces, podemos criar um modelo da interface pretendida e exportá-lo para XML. Depois, a *libglade* converte o ficheiro XML para estruturas em C, criando a interface. Assim, é bastante versátil, a criação e alteração de interfaces, quando em desenvolvimento e testes. Desenvolvida sobre a licença GPL.
- libXML – gestão de ficheiros XML. Permite facilmente criar e navegar em ficheiros XML. A própria libglade utiliza esta biblioteca para aceder à informação inscrita nos ficheiros XML.
- Tesseract – software de reconhecimento de caracteres (OCR), inicialmente desenvolvido pela HP e a Universidade de Las Vegas, libertada como *opensource* sob uma licença Apache v2.0. É utilizado pelo *google,* como base para sistemas de OCR mais completos e complexos, por exemplo o OCRopus – considerado o futuro a nível de OCR.

A utilização de software opensource traz vantagens, não só no custo de aquisição/utilização, mas também na portabilidade e possibilidade de melhoria.

Estas bibliotecas, estão disponíveis para todos os Sistemas Operativos (SOs) existentes, permitindo à aplicação desenvolvida uma utilização idêntica seja em

Microsoft Windows, Linux ou qualquer outro Sistema Operativo (SO).

Além disso, visto que o desenvolvimento foi todo efectuado em C, o Gtk+ surgiu como opção mais aconselhada. Em termos de futuro, o gtk+ tem suporte para todas[1] (!) as linguagens, pelo que torna o desenvolvimento da aplicação mais apetecível, através da utilização de uma outra linguagem mais acessível. O OpenCV está também disponível para outras linguagens, nomeadamente o python- Porém, dado que é uma biblioteca optimizada para velocidade de processamento, não deverá surgir suporte a mais linguagens (visto que muitas linguagens são conhecidas por terem problemas de performance e elas próprias poderem incluir bibliotecas de C).

4.2. Progresso

O trabalho desenvolvido foi desenvolvido em 3 fases principais:

- Fase 1 – Preparação, planeamento e análise de tecnologias existentes;
- Fase 2 – Desenvolvimento;
- Fase 3 – Aperfeiçoamento.

Fase 1

No início desta fase deu-se prioridade à aquisição de competências na programação sobre a linguagem C. Foram efectuados vários programas exemplo de complexidade crescente, que incluíram a biblioteca OpenCV e ambiente gráfico.

Ainda nesta fase, relembraram-se conceitos, nomeadamente sobre visão, o modo de processar imagens e alguns algoritmos de visão. Seguiu-se o teste de algumas aplicações de Visão disponíveis no mercado [...] e o conhecimento de algumas soluções disponíveis neste campo para a industria, nomeadamente câmaras com processamento de imagem integrado ou os módulos integrados sensor/câmara.

Após esta experiência e com os requisitos definidos procedeu-se à realização de uma versão 0 e consequente planeamento da aplicação baseado nesta versão.

Foi definido o nome da aplicação “VAPI”, como acrónimo de “Visão Artificial

1 O gtk está disponível *nativamente* em c, c++, c#, java, python, perl e php, entre outras.

para a Indústria”[2].

Fase 2

Adquiridas competências e planeada a aplicação, deu-se início ao desenvolvimento.

Baseada na versão 0, foi desenvolvida a aplicação com aposta firme na generalização das estruturas de dados, e com objectivo da expansibilidade.

Foram criadas as primeiras “Operações” e definida a relação entre “Tarefas” e “Operações” (ver figura 3.3). À medida que o desenvolvimento avançava houve a necessidade de generalização, principalmente na gestão de opções, associadas às “Operações”. Muito tempo de desenvolvimento foi ocupado na disponibilização de estruturas que possibilitassem versões da aplicação com um mínimo de funcionalidades:

- criação de um modelo de gestão de listas – necessárias para a lista de operações, lista de tarefas e para as próprias tarefas
- disponibilização de um modelo abstracto para a captura de imagem, independente do tipo de fonte (ficheiro, câmara local, câmara remota)
- criação de base para a introdução de novas operações, com a possibilidade de mais facilmente definir as mesmas.

O desenvolvimento a nível da interface teve nesta fase uma grande importância:

- foram desenvolvidas as primeiras verdadeiras interacções com o utilizador onde, por exemplo,
- possibilidade de exportar/importar “Tarefas” para ficheiro, entre outras funcionalidades.

Para demonstrar as funcionalidades foram também criadas tarefas exemplo.

A versão 0.0.4 marcou o fim desta fase, as funcionalidades base estavam estabelecidas, pelo que se fiabilizou a aplicação. Com base nesta versão foram analisados pontos cruciais na aplicação, como a interface e o diálogo com o utilizador, modos e velocidades de execução, bem como opções disponibilizadas.

Nesta fase foi desenvolvida uma caixa de luz controlada, de modo a ser possível efectuar testes com boa repetibilidade, mantendo as condições de luminosidade constantes.

Fase 3

2 Nome sugerido pela colega Ana Graça

Esta fase final ficou marcada por melhorias na aplicação, além do desenvolvimento de toda a componente de comunicação.

No VAPI (Visão artificial para a Indústria) houveram as seguintes evoluções:

- O sistema de opções foi totalmente generalizado, incluindo a importação/exportação de opções para XML, por forma a ser utilizado para o sistema de comunicação.
- uma nova interface, mais funcional e intuitiva.
- A execução de tarefas passou a ser feita de forma paralela (em relação à interface) – utilizando novas "*threads*[3]". Agora na execução das tarefas as estruturas de dados são duplicadas e é possível efectuar alterações às tarefas enquanto elas estão a ser executadas.
- Mais e melhor informação ao utilizador
- criado um agregador de mensagens – de informação e erros.
- Passaram a ser efectuadas mais verificações e melhores verificações, de modo a tornar o sistema mais robusto. Além disso uma melhor gestão de eventos para evitar entrar em concorrência (race condition[4]).

3 Thread – é uma execução paralela à aplicação principal
4 Race Condition – Quando dois eventos querem alterar o mesmo bloco de memória entram em competição sobre quem acede primeiro. Claro que o que chegar em 2º lugar já altera o local de memória errado e a aplicação colapsa.

5. O Vapi

5.1. A Janela Principal

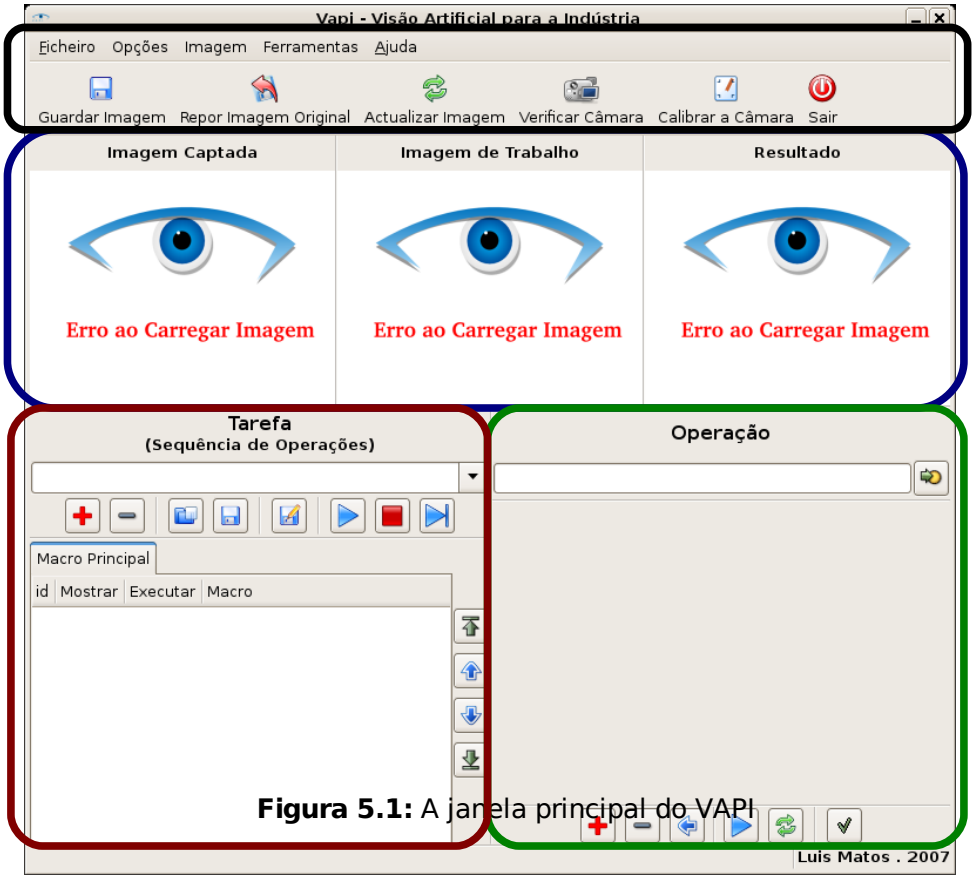

Figura 5.1: A janela principal do VAPI

Figura n: A janela principal do VAPI

A Preto temos o menu e a barra de ferramentas, onde podemos aceder ao menu de escolha da origem da imagem, configurar IO's, calibrar a câmara, entre outros.

A Azul as imagens:

- Captada – a imagem sem alterações
- Trabalho – a imagem sobre a qual se está a trabalhar
- Resultado – o resultado obtido pela aplicação de uma tarefa ou operação

A Verde temos a zona de selecção de operação e definição das opções. Além disso, permite:

- Executar a operação sobre a imagem de trabalho;

- Guardar o resultado como imagem de trabalho;
- Adicionar/Remover a operação actual à tarefa;
- Aplicar as Alterações à operação seleccionada na tarefa.

A Vermelho temos a zona de Tarefas, que permite:

- Criar/Eliminar tarefas
- Abrir/Guardar uma tarefa de/para ficheiro
- Alterar os dados relativos à tarefa
- Executar a tarefa, completa ou passo-a-passo
- Terminar a execução de uma tarefa

5.2. Outras Janelas

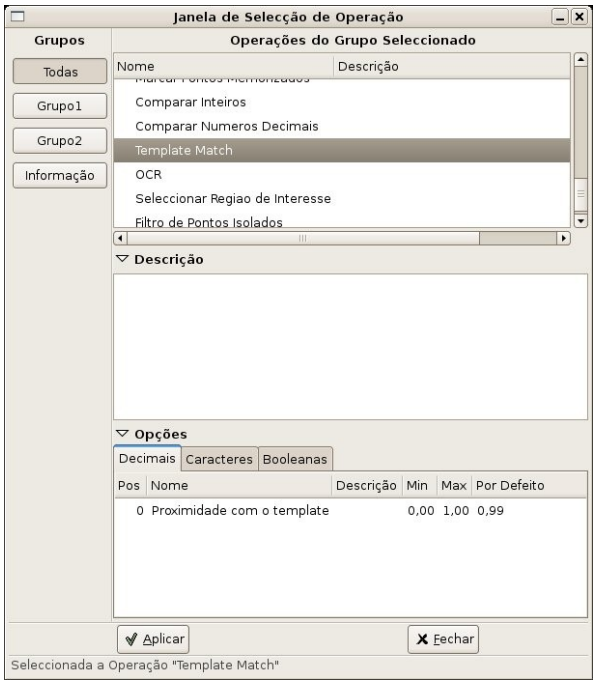

Figura 5.2 – Selecção de Operação

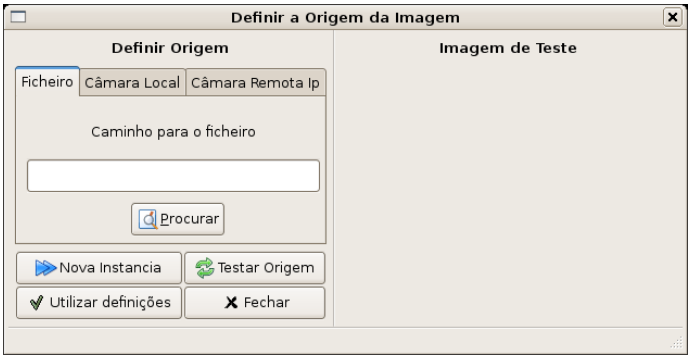

Figura 5.3 – Selecção da Fonte de Imagem

Figura 5.4 – Janela de Comunicações

5.3. Funcionalidades

O VAPI disponibiliza aos utilizadores as seguintes funcionalidades:

- Consegue obter-se resultados em apenas três passos – Definir origem da imagem, seleccionar operação e executar operação;
- Modelo funcional simples: Operações -> Tarefa;
- Rápida execução de uma tarefa;
- Expansível e versátil: é possível desenvolver novas operações e expandir a sua funcionalidade de uma forma bastante simples;
- Não requer que o utilizador tenha muitos conhecimentos, pois a aplicação permite aprendizagem fácil través da utilização de uma interface amigável (“user friendly”).

5.4 Esquemas internos

Pretende-se nesta sub capítulo apresentar um pouco sobre o funcionamento interno do vapi, nomeadamente o seu modelo de início.

Internamente, o VAPI assenta sobre duas funções principais:

- main – carrega todo o baixo nível nomeadamente operações e IO's.
- vapiGtkInterface – é a função main para a interface, que carrega todas as janelas e conecta eventos a sinais.

Depois desta fase, o Gtk controla todos os movimentos. Cada gesto de rato ou tecla pressionada, entre outros, despoleta é considerado um sinal. Cada sinal pode gerar um evento.

Considera-se um evento um qualquer acontecimento. O comum é a cada sinal ser associado a uma função. É essa função que é o evento.

Um exemplo é o click num botão, para preencher uma caixa de texto. É uma função que foi previamente definida que se encarrega da alteração de texto. O Gtk apenas a chama devido à associação que ela possuí com o sinal “Botão pressionado”.

5.4.1 - função main

O vapi é desenvolvido em C. Por isso ao ser iniciado, é executada a função *main*. Esta função é responsável por carregar todos os componentes da aplicação e lançar a interface. O inicío da aplicação é então esquematizado da seguinte forma:

Figura 5.5 – Esquema da função main

5.4.1 função vapiGtkInterface

Esta função tem como objectivo iniciar todas as janelas necessárias ao funcionamento do VAPI.

Por inicialização compreende-se:

- Criação das janelas;
- Carregar a interface das janelas de XML, utilizando a libGlade;
- Definição de elementos das janelas;
- Ligação de sinais a eventos.

6. Desenvolvimento Futuro

É de salientar que a aplicação conta com poucos meses de desenvolvimento quando comparada com aplicações comerciais. Por isso, é necessário na continuação da sua melhoria e, sobretudo, na fiabilização e expansão.

Apresentam-se os seguintes tópicos como sugestões quanto ao futuro da aplicação a nível de desenvolvimento:

- Reorganização de código – de modo a simplificar o desenvolvimento e permitir, por exemplo, a criação de bibliotecas para desenvolvimento;
- Criar bibliotecas para desenvolvimento e carregamento dinâmico de Módulos de "Operações" e Comunicações (no seguimento do ponto anterior);
- Execução de "Tarefas" concorrentes.

  É possível, neste momento, executar uma Tarefa independentemente do que esteja a ser processado na aplicação.

  De modo a que aplicação seja mais versátil, deve ser possível a execução de mais do que uma tarefa simultâneamente.
- Adição de mais tipos de dados às opções

  Devem ser adicionado suporte a mais tipos de dado que possam ser necessários por quem desenvolve operações, tais como:

  - Pontos 2D– que foram emulados utilizando várias opções de valores inteiros.
  - Pontos com coordenadas de euler (3D + orientações), sobretudo para comunicação entre a aplicação e manipuladores.
- Criação de Estruturas de dados para resultados relacionados com cada "Operação" - na aplicação apenas existe uma estrutura de resultados comum a todas as operações pertencentes a uma "Tarefa", não sendo possível aceder directamente aos dados resultantes de cada "Operação".
- Sistema global de gestão de erros que informe melhor o utilizador e previna que a aplicação encerre por acesso indevido localizações de memória.
- Melhorar o interface gráfico de modo a torná-lo mais intuitivo e interactivo com o utilizador.

7. Conclusões

Foi gratificante trabalhar num projecto único como uma aplicação para visão industrial. Como esta aplicação facilita a interligação entre o desenvolvimento de algoritmos e soluções a nível industrial. A utilização de visão em processos industriais fica assim, facilitada.

A maior inovação do VAPI é: a facilidade de desenvolvimento e utilização de algoritmos desenvolvidos e a capacidade inerente, de ser igualmente fácil testar esses algoritmos em situações reais.

As maiores dificuldades prenderam-se com a procura de metodologias de programação correctas e o estudo de problemas concretos.

Os objectivos do projecto foram cumpridos, culminando numa aplicação que:

- Permite a aquisição de várias fontes de imagem;
- é capaz de solucionar problemas simples;
- Tem uma interface amigável (“user friendly”);
- possui uma interface de comunicação com outros sistemas (hardware e software);
- permite a integração de novos algoritmos para processamento de imagem;

Contudo, a aplicação necessita de ser mais testada com o objectivo de ver a sua fiabilidade confirmada.

O VAPI é um novo aliado no desenvolvimento e teste de novos algoritmos, para Visão Artificial e na demonstração das suas capacidades no contexto industrial.

8. Bibliografia Electrónica

1. http://www.gtk.org
2. http://www.glade.org
3. http://opencvlibrary.sf.net
4. http://xmlsoft.org/
5. http://code.google.com/p/tesseract-ocr/
6. http://scentric.net/tutorial/
7. http://www.stack.nl/~dimitri/doxygen/
8. http://www.cs.cmu.edu/~cil/v-source.html

9. Bibliografia impressa

1. Santos, Vítor" - Elementos de Imagem e Visão por Computador", Aveiro, 2006

2. Kerninghan, Brian W.; Pike, Rob - "The Practise of Programing". 1ª Edição. Reading: Addison-Wesley, 1999.

3. Pratt,William K. - "Digital image processing". 3ª Edição. Nova Iorque : John Wiley, 2001.

4. Schalkoff, Robert J. - "Digital image processing and computer vision". Nova Iorque : John Wiley, 1989.

5. Warkus, Matthias; Icaza, Miguel - "The Official Gnome 2 Developer's Guide". No Starch Press, 2004.