Mestrado em Engenharia Informática

Computação Gráfica

Projecto Integrado

Ana Catarina Macedo :: **a54773**
Miguel Palhas :: **pg19808**
Pedro Costa :: **pg19830**

Índice

Contextualização

Este relatório descreve o trabalho realizado para o Projecto Integrado da Unidade Curricular de Computação Gráfica. O trabalho até agora realizado corresponde à componente do projecto referente ao módulo de Visão por Computador.
Esta fase do trabalho foca-se na temática da Realidade Aumentada, para reconhecimento de padrões com auxílio de uma webcam.

O objectivo para esta fase consiste em implementar a detecção de várias marcas em simultâneo, e de seguida obter informação sobre elas.

O programa criado deve analisar o input obtido através da webcam e, dado um conjunto pré-fornecido de informações acerca de padrões bidimensionais, encontrar essas formas no mundo real através da câmara.

Conseguindo implementar a detecção, deve ser também encontrada uma forma de obter informações sobre a posição das marcas encontradas no mundo. Dados como as coordenadas, escala, orientação e rotação da marca detectada, em relação a uma perspectiva inicial, como a do referencial da câmara.

Nas próximas secções é explicada a abordagem usada no projecto, bem como as partes mais importantes do código criado.

Ferramentas Utilizadas

O projecto foi desenvolvido inteiramente em C++, usando o Visual Studio 2010 como ambiente de desenvolvimento.

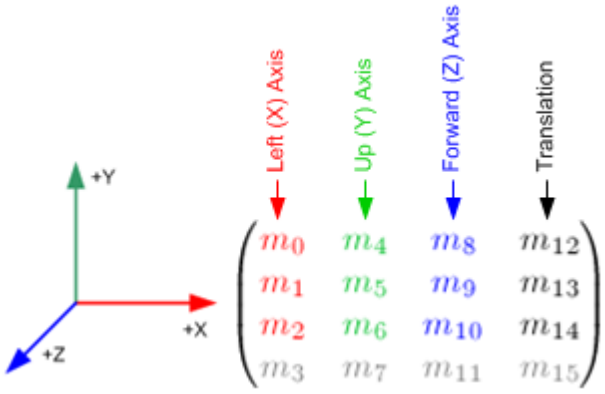
Para auxilio na implementação da detecção de marcas foi usada a biblioteca ARToolkit, uma biblioteca direccionada à criação de aplicações de Realidade Aumentada. O ARToolkit gere toda a parte de detecção das marcas, restando apenas a interpretação dos resultados por ele obtidos.

Extracção de informação

Recorrendo à API da biblioteca ARToolKit é possível obter, para cada marca detectada, a matriz de transformação OpenGL, onde consta informação do seu posicionamento e orientação (em relação à câmara).

Esta matriz pode ser utilizada directamente pela biblioteca OpenGL para definir o

estado do sistema de coordenadas, permitindo colocar automaticamente um objecto sobre a marca. De modo a testar a exactidão do algoritmo de extração de informação, são colocados dois objectos distintos na mesma marca: o primeiro colocado automaticamente carregando a matriz obtida; o segundo colocado manualmente com a posição e orientação extraídas. A comparação dos dois objectos permite, assim, medir a qualidade da informação extraída.



A informação de posicionamento pode ser obtida a partir da matriz de forma trivial: as coordenadas do ponto central da marca correspondem à última coluna. Aplicando estas coordenadas ao objecto colocado manualmente, este adquire exactamente a mesma posição do objecto colocado automaticamente.

A orientação do objecto tem de ser extraída através da resolução de um sistema de equações assente na matriz de rotação apropriada. Assume-se três rotações elementares (uma por eixo), na ordem `RzRyRx`. Para a resolução deste sistema recorre-se à informação da orientação da marca, presente na submatriz de dimensão 3 no canto superior esquerdo da matriz de transformação.

$$
\begin{aligned}
&R_X R_Y R_Z \\
&= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos A & -\sin A \\ 0 & \sin A & \cos A \end{pmatrix} \begin{pmatrix} \cos B & 0 & \sin B \\ 0 & 1 & 0 \\ -\sin B & 0 & \cos B \end{pmatrix} \begin{pmatrix} \cos C & -\sin C & 0 \\ \sin C & \cos C & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos B & 0 & \sin B \\ \sin A \sin B & \cos A & -\sin A \cos B \\ -\cos A \sin B & \sin A & \cos A \cos B \end{pmatrix} \begin{pmatrix} \cos C & -\sin C & 0 \\ \sin C & \cos C & 0 \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} \cos B \cos C & -\cos B \sin C & \sin B \\ \sin A \sin B \cos C + \cos A \sin C & -\sin A \sin B \sin C + \cos A \cos C & -\sin A \cos B \\ -\cos A \sin B \cos C + \sin A \sin C & \cos A \sin B \sin C + \sin A \cos C & \cos A \cos B \end{pmatrix}
\end{aligned}
$$

Os ângulos resultantes da resolução deste sistema (A, B e C) são utilizados com as primitivas de rotação da biblioteca OpenGL. Desta forma, o objecto colocado manualmente fica relativamente próximo ao objecto colocado automaticamente, com algumas flutuações visíveis, causadas provavelmente devido aos erros de arredondamento e ao comportamento periódico das funções trigonométricas.

Implementação

Esta fase foi feita inicialmente com recurso ao código fornecido como exemplo pelo ARToolkit. É já disponibilizado um *sample* que permite fazer a detecção de uma marca previamente definida, usando a sua informação para exibir um cubo na posição da marca.
Este sample foi útil para começar por ter uma melhor compreensão do fluxo de funcionamento do ARToolkit.

Com base neste sample, foi então desenvolvido o código final. A diferença essencial é que o projecto foi implementado em C++, e tira portanto proveito de classes. Existem apenas duas classes:

- **ARWrapper**, colecção de métodos estáticos que servem de interface com o ARToolkit
- **Pattern**, armazena informação relativa a um padrão. Uma lista de instâncias desta classe serve de base ao algoritmo de pesquisa para detecção dos padrões em cada imagem capturada.

Eis o bloco de código que faz a detecção de padrões a cada frame recebida pela câmara:

```
arDetectMarker(dataPtr, patt_thresh, &marker_info, &marker_num);

arVideoCapNext();
num_detected = 0;
// for each marker detected, check for visibility
for (i = 0; i < marker_num; ++i ) {
	// compare current marker with patterns
	for (vector<Pattern>::iterator it = patterns.begin(); it !=
patterns.end(); ++it) {
		if (it->match(marker_info[i])) {
			++num_detected;
			markerDetected(*it, marker_info[i]);
		}
	}
}
argSwapBuffers();
```

Parte da lógica da aplicação está encapsulada na classe Pattern, que gere os padrões pré-registados para detecção pelo ARToolkit. O método markerDetected é invocado para cada

padrão detectado, e é o responsável por efectuar operações sobre a sua informação, como por exemplo, fazer o render de um objecto na posição do padrão, ou extrair dados relativos á posição e orientação do padrão no mundo.