ChatGPT

Arquitetura completa do pipeline de vídeo

O pipeline proposto segue quatro etapas principais: **roteiro/storyboard**, **geração de keyframes**, **produção de vídeo** e **pós-produção**. Na fase de roteiro (pré-produção), geralmente um modelo de linguagem (ou "planner") interpreta a ideia geral e gera um *storyboard* ou roteiro de cenas estruturado [1]. Em seguida, cada cena definida no storyboard é convertida em *keyframes* – imagens âncora que representam o início e o fim de cada fragmento de vídeo. Por exemplo, ferramentas modernas de IA conseguem transformar roteiros textuais em planos visuais: "softwares avançados rapidamente transformam scripts em planos visuais; a IA interpreta palavras para gerar keyframes, posicionamento de personagens e cenários" [2]. A função `run_keyframe_generation()` executa essa etapa, produzindo imagens fixas ("Big Bang" e "Big Freeze") que servirão de pontos de contorno para a animação [3].

Com os keyframes definidos, entra a **produção de vídeo**: um modelo de vídeo (por exemplo, um modelo de difusão temporal) recebe como entrada o keyframe inicial, o keyframe final, um prompt de transição e o contexto causal anterior, para gerar o fragmento de vídeo correspondente [4]. Por fim, na **pós-produção**, fragmentos consecutivos são unidos. Implementações usuais usam FFmpeg para aparar as últimas frações sobrepostas de cada clipe (evitando "glitch" visual) e em seguida concatenar todas as partes em um vídeo final coeso [5]. Em código, isso pode ser feito por uma função como `concatenate_and_trim_masterpiece()`, que realiza justamente esse corte e concatenação dos fragmentos finais [6].

- **Roteiro/Storyboard (pré-produção):** define a sequência de cenas e keyframes. Um modelo "planner" (chamado de *Gemini* na implementação) lê a ideia geral e produz o *storyboard* ("scene_storyboard"), determinando a ordem lógica e os keyframes de cada cena [1].
- **Geração de keyframes:** cada cena ganha keyframes iniciais e finais fixos. Uma ferramenta de geração de imagem (por exemplo, *DreamO*) materializa essas âncoras visuais. A função `run_keyframe_generation()` cria imagens estáticas que representam o começo e o fim do fragmento de vídeo [3]. Essas imagens definem o "Big Bang" e o "Big Freeze" de cada segmento, garantindo pontos de partida e chegada claros para o modelo de vídeo.
- **Produção de vídeo:** um modelo de geração de vídeo (por exemplo, *LTX*) faz uma interpolação controlada entre os keyframes, condicionada pelo prompt narrativo e pelo contexto anterior. Ele recebe como entrada o keyframe inicial, o keyframe final, um prompt de transição sintetizado e o contexto de "eco" herdado do fragmento anterior, produzindo o clipe de vídeo correspondente.
- **Pós-produção:** fragmentos gerados são concatenados. Primeiro aparando as sobreposições finais (últimos quadros) de cada fragmento (exceto o último), depois concatenando todos os clipes em um único vídeo de saída [5] [6]. Isso remove artefatos visuais e unifica os segmentos em uma obra final contínua.

Memória cinética ("eco") e `kinetic_memory_path`

Para manter a continuidade entre os fragmentos, cria-se uma "memória cinética" ou *eco* de cada vídeo gerado. Na prática, isso significa extrair os últimos N quadros de um fragmento e salvá-los como um pequeno clipe de vídeo. Este clipe de "eco" carrega a **memória física** (velocidades, movimentos,

enquadramentos) do final do clipe anterior e serve como condicionante inicial no próximo. Como descrito na implementação teórica, “o conceito abstrato do ‘eco’ é materializado como um clipe de vídeo curto. A função de extração usa FFmpeg para contar frames e extrair os últimos *CONVERGENCE_FRAMES* do fragmento recém-gerado. Este clipe, salvo em disco, torna-se o portador da ‘memória física’ e do ‘vetor de inércia’” [7] .

Em código, isso corresponde à linha:

```
kinetic_memory_path = extract_last_n_frames_as_video(...)
eco_output_path = os.path.join(WORKSPACE_DIR,
f"eco_from_frag_{fragment_num}.mp4")
```

ou seja, cria-se um vídeo curto (via FFmpeg) contendo os quadros finais (`extract_last_n_frames_as_video`) e define-se um caminho de saída (`eco_output_path`) para armazenar esse “eco”. Esse clipe de eco sobrevive ao descarte de memória GPU e é injetado como contexto causal no próximo fragmento, garantindo transições mais suaves e fisicamente coerentes [7] .

Decisão de Transição (cut vs contínuo)

Na edição cinematográfica, a transição entre cenas pode ser feita de modo **abrupto** (corte seco) ou **suave** (transição contínua como fade, movimento de câmera, etc.). Em termos narrativos, “a transição é uma forma de ir de uma cena para outra sem que haja percepção abrupta de mudança... Bem usadas, as transições contribuem para a construção da narrativa” [8] . Já um corte simples (*cut*) troca diretamente a cena sem efeito adicional. Ambos têm usos criativos diferentes: cortes podem acelerar o ritmo ou surpreender, enquanto transições contínuas preservam fluidez e ritmo emocional.

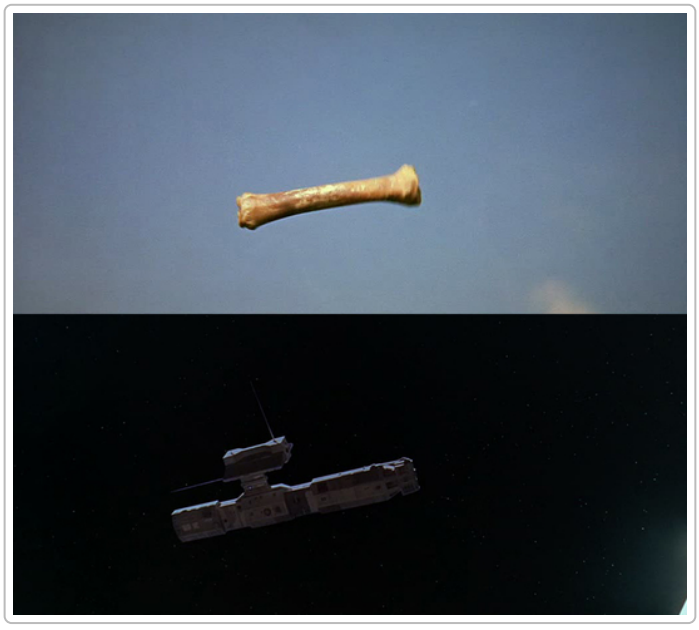

Figura: Exemplo do match cut *icônico do filme "2001: Uma Odisseia no Espaço" (1968), ilustrando uma transição cinematográfica entre dois objetos (um osso e uma nave espacial) que mantém uma continuidade visual temática* [9] . O *match cut* acima é um caso famoso de transição contínua (suave) baseada em similaridade visual. Em contraste, um simples **cut** interromperia bruscamente a cena.

Na prática do pipeline, há uma função dedicada que decide, para cada transição entre fragmentos, se será utilizado um corte seco ou uma passagem contínua. Essa decisão pode ser delegada a um modelo de IA: por exemplo, uma chamada a um LLM que avalia o roteiro, o histórico da história e até a "memória" passada (arquivo de eco), retornando a escolha. O código `get_transition_decision(user_prompt, story_history, memory_media_path, ...)` exemplifica isso: ele consulta a IA, que responde em JSON se a transição deve ser *"cut"* ou *"continuous"*. Em termos de implementação, é como pedir que o modelo gere uma instrução de edição. Exemplos de prompts de transição suave seriam, por exemplo, "a câmera desacelera e foca no rosto do personagem", que orientam o modelo de vídeo a interpolar suavemente entre cenas [10].

Em síntese, os **tipos de transição** comuns são:
- **Corte (Cut):** troca imediata entre cenas, sem efeito visual adicional. Pode ser usado para ritmo acelerado ou impacto súbito.
- **Contínuo:** inclui dissolvências, movimentos de câmera (pan, tilt), *match cut* e outras técnicas que unem duas cenas de forma suave, preservando a continuidade de movimento ou temática [8].

Pesquisas recentes mostram que gerar vídeos longos com várias cenas requer atenção explícita a essas transições – modelos genéricos tendem a produzir apenas uma cena contínua se não forem treinados para detectar mudanças no prompt [11]. Nossa função de decisão de transição permite contornar isso, aplicando o tipo de corte mais apropriado conforme a narrativa exigida.

Conceitos separados no código

Os três conceitos destacados são de fato tratados de forma separada no pipeline:

- **Pipeline de produção (roteiro → keyframes → vídeo → pós-produção):** implementado pelas funções `run_storyboard_generation()`, `run_keyframe_generation()` e `run_video_production()`. Cada uma executa uma fase distinta do fluxo. Conforme visto, o módulo de planejamento (*Gemini*) cuida do storyboard e keyframes [1], enquanto o modelo de geração de imagens aplica `run_keyframe_generation()` para materializar as âncoras visuais [3]. A concatenação final é feita após a produção completa de todos os fragmentos [6].
- **Memória cinética / "eco":** implementada pela extração de quadros finais e salvamento em arquivo (`kinetic_memory_path` e `eco_output_path`). Essa função é responsável apenas por capturar o clipe de eco como memória física do fragmento anterior, sem misturar lógica de transição ou geração de cena [7].
- **Decisão de transição ("cut" vs "continuous"):** realizada na função `get_transition_decision()`. Ela não interfere na geração de keyframes ou vídeo em si, apenas escolhe o estilo de passagem entre cenas com base no prompt, histórico e memória disponível. Ou seja, é um componente dedicado à lógica de edição, separado da geração de conteúdo visual em si.

Cada bloco de código atua de forma isolada em seu domínio. A arquitetura geral é modular: planejamento de cenas, geração de keyframes e vídeo, extração de memória (eco) e decisão de transição são processos independentes, mas integrados no fluxo final. Essas separações tornam o sistema mais organizado e cada etapa pode ser aprimorada ou substituída sem afetar diretamente as outras [1] [7].

Fontes: literatura sobre pipelines de produção de vídeo e storyboards automáticos [2] [1]; trabalho acadêmico que implementa conceitos de memória causal e transição em geração de vídeo [10] [7];

artigos de cinema sobre o papel das transições na narrativa [8] [9] ; estudos recentes sobre geração de vídeo de várias cenas [11] .

---

[1] [3] [4] [5] [6] [7] [10] ADUC-SDR_Thesis.pdf
file://file-XmQHhNvAnQakRgRodDGvXx

[2] AI Storyboard Generator from Script for Fast Video Production | HeyGen
https://www.heygen.com/blog/ai-storyboard-generator-from-script

[8] [9] A Importância da Transição | Blog da AvMakers
https://www.avmakers.com.br/blog/a-importancia-da-transicao

[11] Enhancing Scene Transition Awareness in Video Generation via Post-Training
https://arxiv.org/html/2507.18046v1