Objectif

- Mettre en place et évaluer une pipeline **image → texture** sur un mesh de canard prédéfini (UV).
- Comparer les outputs Phase 1 (**FLUX** vs **GPT-5**) comme entrée de la Phase 2 (**SDXL image-to-texture**).
- Vérifier la qualité sur le mesh via multiview (coutures UV, distorsions, cohérence) via benchmark.

Contraintes

Données / mesh

- Mesh canard + UV
- Une seule image de référence (single view)
- Les zones cachées doivent être "inventées" → erreurs fréquentes

Génération d'image (Phase 1)

- Risque de changer la silhouette du canard
- Si la forme change → artefacts en projection texture

On-premise / GPU

- Pipeline Phase 2 très lourd (SDXL)
- Besoin > 32 GB VRAM pour être confortable
- Le coût GPU limite le volume de tests

Robustesse texture (Phase 2)

- Non déterministe (même input → résultats différents)
- Fuite de background sur la texture

Outils et technologies

Phase 1 — Image generation

- FLUX Kontext (local)
- GPT-5 image diffusion (rapide, meilleur rendu)
- Benchmark : 100 images / modèle

Évaluation

- Multiview renders (face, côtés, dos, top, down)
- Viewer 3D pour inspection du `.glb`
- Grilles : contraintes couleur + alignement + cohérence

Phase 2 — Image-to-texture

- Pipeline basé sur SDXL
- Input : image Phase 1
- Output : texture + `.glb` texturé + multiview auto

Infrastructure

- Exécution GPU (A100 dans les tests)
- FLUX 3 min / image
- GPT-5 20 s / image
- Automatisation possible (batch runs) si GPU dispo

Échecs typiques

- Background leakage : fond transféré sur la texture.
- Silhouette change : forme modifiée en Phase 1 → projection incohérente.
- UV stretching : cassures et formes étirées.
- Single view hallucination : arrière inventé → incohérences multiview.

Observations clés

- GPT-5 respecte mieux le design et la forme globale du canard.
- Résultats "acceptables" : 58% GPT-5 vs 18% FLUX (sur 100 générations).
- Les contraintes couleur sont globalement bien suivies (surtout couleurs plates).
- Les gros échecs viennent de : silhouette modifiée, background qui fuit, incohérences multiview.
- La Phase 2 amplifie les erreurs de la Phase 1 : si l'image est "bizarre", la texture casse vite.

Scalabilité

La pipeline est simple (Phase 1 → Phase 2), donc elle peut scaler en volume. Mais en pratique, le bottleneck est GPU : Phase 2 est coûteuse (>32 GB VRAM) et non déterministe. FLUX est on-premise mais trop lent. GPT-5 est rapide mais pas on-premise. FLUX arrête sa génération en cas de copyright.

Apprentissage

Avec une seule image, le modèle doit deviner l'arrière du mesh. Donc il hallucine. Ça crée répétitions, distorsions et incohérences entre vues. Pour réduire ça, il faut plus d'info en entrée (multi-vues) ou une méthode plus "geometry-aware".

Décision

La pipeline marche parfois, mais pas assez fiable pour la production. Pour améliorer : il faut retirer le background (segmentation) avant Phase 2, faut forcer la préservation de la silhouette en Phase 1, faut utiliser plusieurs vues en entrée au lieu d'une seule, faut tester des méthodes plus orientées texture + géométrie (meilleure cohérence multi-vues).

---

1. Analyse des résultats Image-to-Texture

1.1. Vue d'ensemble du pipeline

On a une pipeline en 2 phases.

La phase 1 crée une image 2D du canard. La phase 2 transforme cette image en texture UV sur notre mesh.

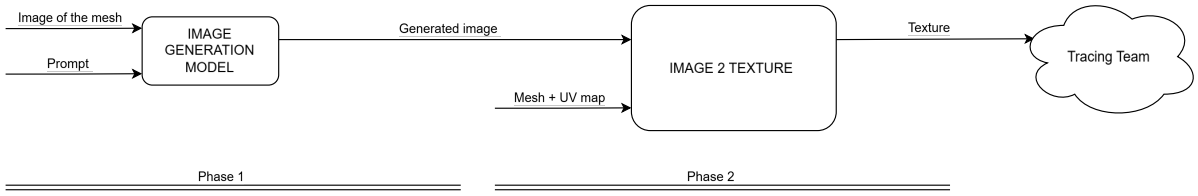

Figure 1: Pipeline image ⟶ texture (2 phases).

Phase 1 — On génère un rendu 2D du canard avec le design demandé.

Input : image de référence

Output : image 2D du canard (design)

Elle sert d'entrée pour la phase 2.

Phase 2 — On prend l'image générée en phase 1 et on la projette sur le mesh 3D.

Input : image de la phase 1

Mesh : canard (.glb)

Output : texture

On évalue ensuite le mesh texturé en multiview.

1.2. Évaluation multiview

Pour vérifier la cohérence, on rend le mesh depuis plusieurs angles.

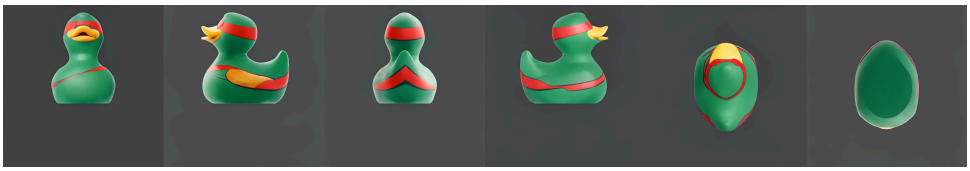

Figure 2: Multiview (face, 2 côté, dos, top, down)

Ce qu'on regarde :

coutures UV visibles

stretching (formes étirées)

incohérences entre vues

1.3. Alignement texture / géométrie

On vérifie si les éléments du design tombent au bon endroit :

Figure 3: Mesh texturé dans le viewer 3D.

On fait attention aux zones “compressées” dans l’UV. C’est souvent là que ça casse.

1.4. Contraintes de design

max 4 couleurs

couleurs pleines (flat)

formes simples et lisibles

1.5. Grille d’évaluation

Critère	Résultat	Commentaire
Limite de couleurs respectée	Oui / Non	
Couleurs pleines uniquement	Oui / Non	
Lisibilité des formes	Bon / Moyen / Mauvais	
Alignement sur le mesh	Bon / Moyen / Mauvais	
Cohérence multiview	Bon / Moyen / Mauvais	
Faisable en peinture manuelle	Bon / Moyen / Mauvais	

Table 1: Critères pour juger une texture générée.

1.6. Résultats phase 1

Méthode : FLUX (Black Forest Labs, variante Kontext)

Prompt (exemple) :

““Apply the following design to the duck: Body and head: solid green. Beak: solid yellow. Paint a simple red pirate band across the head where the eyes are located. The band must be a flat painted stripe only and must not include a knot or cloth pieces. Paint a simple red diagonal stripe across the body to represent a pirate sash.””

Mesh cible : canard validé (.glb)

Figure 4: Canard dans le viewer 3D (mesh).

Output :

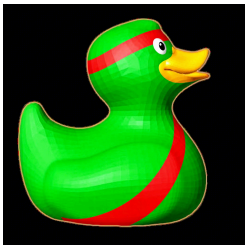

Figure 5: Image générée avec FLUX Kontext.

1.7. Analyse phase 1

On a testé deux systèmes :

FLUX Kontext (local)

GPT-5 image diffusion

En pratique :

FLUX tourne en local, mais c’est lent (≈ 3 min / image sur A100).

GPT-5 est beaucoup plus rapide (≈ 20 s / image).

Benchmark :

100 générations FLUX

100 générations GPT-5

Évaluation : inspection humaine avec la grille définie plus haut.

Exemple GPT-5 :

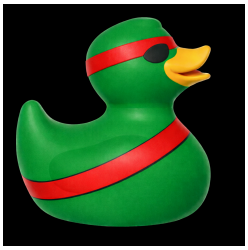

Figure 6: Image générée avec GPT-5.

Globalement, GPT-5 respecte mieux :

le design demandé la cohérence visuelle la forme du canard Résultats pour FLUX :

Critère	Résultat	Commentaire
Limite de couleurs respectée	Oui / Non	Non
Couleurs pleines uniquement	Oui / Non	Oui
Lisibilité des formes	Bon / Moyen / Mauvais	Mauvais
Alignement sur le mesh	Bon / Moyen / Mauvais	Mauvais
Cohérence multiview	Bon / Moyen / Mauvais	Mauvais
Faisable en peinture manuelle	Bon / Moyen / Mauvais	Moyen

Table 2: Résultats phase 1 — FLUX Kontext.

Résultats pour GPT-5 :

Critère	Résultat	Commentaire
Limite de couleurs respectée	Oui / Non	Oui
Couleurs pleines uniquement	Oui / Non	Oui
Lisibilité des formes	Bon / Moyen / Mauvais	Bon
Alignement sur le mesh	Bon / Moyen / Mauvais	Bon
Cohérence multiview	Bon / Moyen / Mauvais	Moyen
Faisable en peinture manuelle	Bon / Moyen / Mauvais	Moyen

Table 3: Résultats phase 1 — GPT-5.

Remarque : même avec un prompt qui demande de garder la forme, certaines images changent la silhouette du canard. Ensuite, en phase 2, ça crée des artefacts à la projection.

Autre remarque : GPT-5 ajoute parfois des détails “logiques” non demandés (ex : un cache-œil pirate). C’est cohérent sémantiquement, mais pas toujours souhaité.

Bilan phase 1 :

textures “acceptables” ≈ 58% avec GPT-5

textures “acceptables” ≈ 18% avec FLUX

Définition : “acceptable” = au moins 4 critères sur 6 en Bon ou Oui.

1.8. Benchmark V2

On a ajouté une grille plus claire, pour être lisible par quelqu'un hors équipe.

Critère	Résultat	Question d'évaluation
Préservation de la forme	Bon / Moyen / Mauvais	Est-ce que la silhouette du canard reste identique ?
Respect du prompt	Bon / Moyen / Mauvais	Est-ce que le design suit bien les instructions ?
Limite de couleurs	Oui / Non	Est-ce qu'on reste dans les couleurs autorisées ?
Couleurs pleines	Oui / Non	Pas de gradients / textures ?
Plausible en peinture	Bon / Moyen / Mauvais	Ça ressemble à de la peinture sur surface ?
Qualité image	Bon / Moyen / Mauvais	Rendu propre ?
OK pour phase 2	Oui / Non	Utilisable pour la conversion texture ?

Table 4: Benchmark V2 — évaluation phase 1 uniquement.

1.9. Résultats phase 2

En phase 2, on prend l'image validée en phase 1 et on lance l'image-to-texture.

Le pipeline est basé sur SDXL.

Points importants :

modèle non déterministe (même input ⟶ résultats différents)

pipeline lourd (besoin GPU élevé) dans notre setup, il faut plus de 32 GB de VRAM

Outputs :

un .glb texturé des renders multiview automatiques

Exemple Flux qualité :

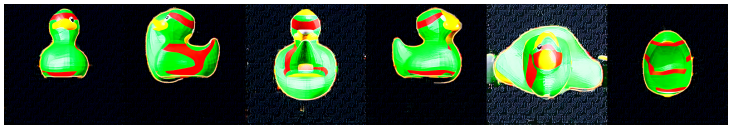

Figure 7: Exemple — résultat multiview faible.

Exemple GPT qualité :

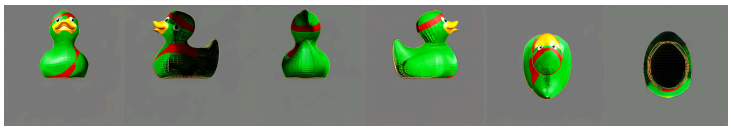

Figure 8: Exemple — résultat multiview bon.

Méthode d'évaluation :

ouvrir le .glb dans un viewer 3D

vérifier plusieurs angles

compléter avec des captures manuelles

Cas d'échec fréquent :

le background de l'image "fuit" dans la texture

déformations locales

éléments mal placés

incohérences entre faces

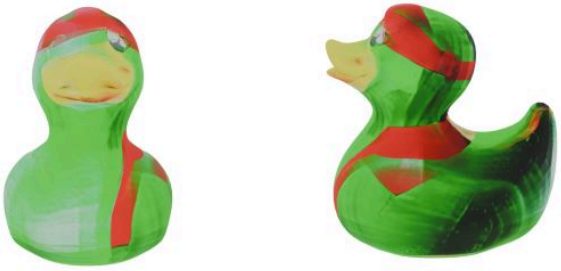

Figure 9: Exemple — rendu final faible.

Figure 10: Exemple — rendu final bon.

Grille d'évaluation phase 2 :

Critère	Résultat	Question
Alignement texture / mesh	Bon / Moyen / Mauvais	Les éléments suivent bien les zones du canard ?
Artefacts de coutures UV	Faible / Moyen / Fort	On voit des cassures aux coutures ?
Distorsion texture	Faible / Moyen / Fort	Formes étirées / compressées ?
Cohérence multiview	Bon / Moyen / Mauvais	Le design reste cohérent selon l'angle ?
Fuite de background	Oui / Non	Des couleurs du fond apparaissent sur le mesh ?
Placement des éléments	Bon / Moyen / Mauvais	Bandes / stripes au bon endroit ?
Peinture manuelle faisable	Bon / Moyen / Mauvais	Reproductible à la main ?
OK pour tracing	Oui / Non	Utilisable par l'équipe tracing ?

Table 5: Critères phase 2 — résultat texture final.

1.10. Limites observées

On a 3 limites principales.

Limite 1 — On-premise On doit tourner en local. Donc on favorise des modèles open-weight.

Limite 2 — FLUX dans notre cas Sur image-to-texture, on voit souvent :

artefacts incohérences entre vues background qui se transfère sur le canard

Figure 11: Pipeline complet et render multiview (exemple).

fig:full-pipeline

Souvent, la vue avant est OK. Mais quand on tourne, ça casse.

Figure 12: Même prompt, variantes FLUX Kontext.

fig:flux-variations

Limite 3 — Single view Le problème vient aussi du format :

une seule image pour couvrir tout le mesh Donc le modèle doit “inventer” l’arrière et les zones cachées. Ça crée :

répétitions incohérences mauvais alignements

1.11. Conclusion

L’approche marche parfois. On obtient des résultats visuellement bons sur certains cas.

Mais la pipeline n’est pas assez fiable pour une utilisation pratique.

Le point le plus bloquant est la robustesse :

phase 1 : FLUX est trop irrégulier (GPT-5 est meilleur mais pas on-premise) phase 2 : la projection crée des fuites de background et des incohérences multiview Pour obtenir un résultat stable, il faudra probablement :

un modèle vraiment orienté texture geometry-aware une meilleure cohérence multi-vues ou plus d’info en entrée (plusieurs vues au lieu d’une seule)

1.12. Décision

Cette méthode n’est pas fiable.