MIROIR VIDÉO

Version 3.7.0

DOCUMENTATION TECHNIQUE

PARTIE 1 : SPÉCIFICATIONS FONCTIONNELLES GÉNÉRALES

1.1 Présentation du Projet

1.1.1 Contexte

Dans la pratique du tir à l'arc, l'analyse du geste technique est un élément fondamental de la progression. Historiquement, les archers s'appuyaient sur le regard extérieur d'un entraîneur ou sur des enregistrements vidéo a posteriori.Ces méthodes présentent des limitations : le retour d'expérience vocal n'offre pas de référence visuelle immédiate, tandis que l'analyse vidéo différée rompt la continuité de l'entraînement.

L'application **Miroir Vidéo** répond à ce besoin en fournissant un **retour visuel temps réel différé** : l'archer voit son geste s'afficher sur un écran quelques secondes après l'avoir exécuté, créant ainsi une boucle d'auto-correction immédiate. Cette approche s'inspire du principe du "miroir différé" utilisé dans d'autres disciplines sportives (gymnastique, danse, arts martiaux).

> 💡 **Cas d'usage typique :**
> Un archer positionne une webcam face à son pas de tir, lance l'application sur un ordinateur portable et observe son geste complet (de la prise de flèche jusqu'à la décoche) affiché avec 7 secondes de décalage sur un écran placé à côté. Cette visualisation immédiate permet d'identifier et de corriger des défauts techniques en temps réel, sans nécessiter de matériel vidéo complexe ni de post-production.

1.1.2 Objectifs du Projet

- **Objectif Principal :** Fournir un outil d'auto-analyse gestuelle accessible, gratuit et sans installation pour les pratiquants du tir à l'arc
- **Objectifs Secondaires :**
  - Simplifier l'entraînement en solo sans nécessiter de coach
  - Permettre une analyse instantanée du geste complet (3-11 secondes)
  - Garantir la confidentialité totale (traitement 100% local)
  - Assurer une compatibilité multiplateforme (Windows, Mac, Linux, mobile)
  - Offrir une expérience utilisateur sans friction (zéro installation, un clic pour démarrer)

1.1.3 Périmètre Fonctionnel

✅ Fonctionnalités Incluses

- Capture vidéo webcam en temps réel
- Affichage miroir avec délai configurable (3 à 11 secondes)
- Grille de visée (règle des tiers) avec 7 couleurs au choix
- Changement de source caméra à chaud

- Mode plein écran automatique
- Interface multilingue (français/anglais)
- Raccourci clavier Ctrl+Q pour fermer

❌ **Hors Périmètre (V3.7)**

- Enregistrement ou sauvegarde de vidéos
- Capture d'images fixes (screenshots)
- Analyse automatique du geste (IA/ML)
- Mode comparaison côte-à-côte
- Diffusion réseau ou streaming
- Support audio (son désactivé)
- Annotations ou dessins sur l'image

1.2 Architecture Système

1.2.1 Vue d'Ensemble

L'application adopte une **architecture 100% client-side**, éliminant tout besoin de serveur backend. Elle repose sur les technologies Web standards (HTML5, CSS3, JavaScript ES6+) et s'exécute directement dans le navigateur de l'utilisateur.

> **Principes architecturaux clés :**
> Autonomie totale : Aucune dépendance externe, aucune connexion Internet requise après chargement
> Monolithique : Application contenue dans un unique fichier HTML (~1200 lignes)
> Stateless : Pas de persistance de données entre sessions
> Event-driven : Réactivité basée sur la boucle requestAnimationFrame

1.2.2 Composants Techniques

A. Couche de Capture Vidéo

- **API utilisée :** `navigator.mediaDevices.getUserMedia()`
- **Contraintes :** Résolution 1280x720 (HD) idéale, fallback vers 640x480 si indisponible
- **Flux :** MediaStream connecté à un élément <video> HTML5
- **Gestion caméras multiples :** Énumération via enumerateDevices(), sélection par deviceId

B. Pipeline de Traitement Temporel

1. **Capture frame :** Extraction de l'image courante de l'élément video vers un canvas buffer
2. **Extraction pixels :** Conversion en ImageData via getImageData()
3. **Horodatage :** Association d'un timestamp précis (Date.now()) à chaque frame
4. **Stockage temporaire :** Ajout dans une pile FIFO (frameStack) avec nettoyage des frames périmées
5. **Calcul du délai :** Recherche de la frame correspondant à now - delayMs
6. **Affichage :** Rendu sur le canvas de sortie via putImageData()

> ⚠ **Point critique :**
> Le buffer temporel peut consommer jusqu'à 500 MB de RAM pour un délai de 11 secondes à 720p.
> Cette empreinte mémoire est proportionnelle au produit : résolution × FPS × délai.

C. Interface Utilisateur

Composant	Rôle	Technologie
Splash Screen	Écran d'accueil avec activation manuelle	HTML + CSS (position:fixed overlay)
Canvas Output	Zone d'affichage principal plein écran	Canvas 2D API
Menu Configuration	Fenêtre modale pour paramétrage	Div flottante, événements onclick
Grille de Visée	Overlay 3×3 (règle des tiers)	Canvas strokeRect + 7 couleurs
Barre d'Infos	Affichage des paramètres actifs	Div en bas d'écran

Loader	Indicateur pendant phase de buffering	Animation CSS + compteur JavaScript

1.3 Exigences Techniques

1.3.1 Compatibilité Navigateurs

Navigateur	Version Minimale	Statut
Google Chrome / Chromium	92+ (juillet 2021)	✅ Recommandé
Microsoft Edge	92+ (juillet 2021)	✅ Recommandé
Mozilla Firefox	90+ (juillet 2021)	✅ Supporté
Safari (macOS)	15+ (septembre 2021)	✅ Supporté
Safari (iOS)	15+ (septembre 2021)	⚠ Limité (pas de fullscreen)

Limitations connues : i
OS Safari : L'API Fullscreen n'est pas supportée sur iPhone (fonctionne sur iPad)
Anciens navigateurs (<2021) : API getUserMedia incompatible ou contraintes vidéo non prises en charge
Mode navigation privée : Certains navigateurs bloquent l'accès caméra par défaut

1.3.2 Performances

Critère	Valeur cible
Fréquence d'images	30 fps (images par seconde)
Résolution vidéo idéale	1280x720 pixels (HD)
Latence système	< 50 ms (hors délai configuré)
Temps de démarrage	< 3 secondes (caméra prête)
Utilisation mémoire	< 500 MB (pour délai de 11s)

1.3.3 Sécurité et Confidentialité

Garanties de confidentialité :
Aucune donnée n'est transmise sur Internet Traitement 100% local dans le navigateur
Aucun enregistrement ou stockage des vidéos Pas de cookies ou de pistage utilisateur
Demande explicite d'autorisation pour l'accès à la caméra

1.4 Interfaces Externes

1.4.1 Interface Utilisateur

- **Écran d'accueil (Splash Screen) :** Présentation et activation manuelle
- **Canvas de sortie :** Affichage plein écran du flux vidéo miroir
- **Menu de configuration :** Fenêtre modale pour paramétrage
- **Barre d'informations :** Affichage des paramètres actifs

1.4.2 API Navigateur Utilisées

- `navigator.mediaDevices.getUserMedia()` - Accès caméra
- `navigator.mediaDevices.enumerateDevices()` - Liste périphériques
- `requestAnimationFrame()` - Boucle de rendu
- `Canvas 2D API` - Manipulation d'images
- `Fullscreen API` - Mode plein écran

1.4.3 Dépendances

> 💡 **Aucune dépendance externe :**
> L'application est entièrement autonome et ne nécessite aucune bibliothèque tierce.
> Tout le code est contenu dans un unique fichier HTML avec CSS et JavaScript embarqués.

1.5 Exigences Non Fonctionnelles

1.5.1 Utilisabilité

- **Simplicité :** Interface minimaliste avec un maximum de 3 clics pour toute action
- **Intuitivité :** Pas de manuel nécessaire pour une première utilisation basique
- **Feedback visuel :** Chaque action utilisateur génère une réponse visuelle immédiate
- **Accessibilité :** Compatible avec les écrans tactiles et les interfaces clavier

1.5.2 Fiabilité

- **Disponibilité :** Fonctionnement 24/7 sans serveur (100% client-side)
- **Tolérance aux pannes :** Gestion des erreurs caméra avec messages explicites
- **Récupération :** Redémarrage automatique du flux en cas de déconnexion caméra

1.5.3 Performance

- **Temps de réponse :** Changement de paramètre appliqué en moins de 1 seconde
- **Fluidité :** Maintien de 30 fps même sur matériel modeste (PC de 2015+)
- **Consommation CPU :** Maximum 40% sur un cœur moderne

1.5.4 Maintenabilité

- **Code commenté :** Documentation inline extensive en français
- **Architecture modulaire :** Fonctions clairement séparées et nommées
- **Version centralisée :** Variables CSS pour version et copyright

1.6 Cas d'Usage Métier

1.6.1 Scénario 1 : Entraînement Solo

Acteur : Archer amateur s'entraînant chez lui
Contexte : Préparation d'une compétition, besoin d'améliorer la régularité du geste
Déroulement :

7. Installation du PC portable avec webcam face au pas de tir
8. Lancement de l'application, autorisation caméra
9. Réglage du délai à 7 secondes (tir complet visible)
10. Exécution de 10 tirs en observant chaque geste après coup
11. Identification d'un défaut récurrent (ex: épaule qui remonte)
12. Concentration sur la correction pendant 10 nouveaux tirs
13. Validation de l'amélioration par comparaison visuelle

Bénéfices : Progression technique mesurable, autonomie totale, coût zéro

1.6.2 Scénario 2 : Séance de Coaching

Acteurs : Entraîneur + archer en cours individuel
Contexte : Correction d'un défaut technique identifié lors d'une compétition
Déroulement :

14. Installation de l'écran en grand format (TV ou vidéoprojecteur)
15. Positionnement caméra de profil à 3 mètres
16. L'archer effectue un tir "témoin"
17. Coach et archer analysent ensemble le geste affiché avec 7s de décalage
18. Le coach explique la correction attendue
19. L'archer refait un tir en appliquant la consigne
20. Comparaison immédiate avant/après

21. Répétition jusqu'à intégration du bon geste

Bénéfices : Support visuel pour la pédagogie, validation objective de la progression

1.6.3 Scénario 3 : Installation Permanente en Club

Acteur : Club équipant une salle d'entraînement
Contexte : Mise à disposition d'un outil d'auto-analyse pour tous les licenciés
Déroulement :

22. Installation fixe : webcam HD + PC dédié + écran 50"
23. Application configurée en démarrage automatique
24. Chaque archer arrive, clique pour lancer, s'entraîne
25. Utilisation libre sans réservation ni accompagnement
26. Chacun adapte les paramètres (caméra, délai) selon ses besoins

Bénéfices : Investissement unique, maintenance nulle, usage illimité

1.7 Glossaire

Terme	Définition
Buffer / Buffering	Zone mémoire temporaire stockant les images vidéo avec leur horodatage. Phase de remplissage initial avant affichage.
Canvas	Élément HTML5 permettant le dessin et la manipulation d'images pixel par pixel via JavaScript.
Délai / Delay	Décalage temporel entre la capture d'une image et son affichage, exprimé en secondes (3 à 11s).
Frame / Image	Une image fixe capturée à un instant T depuis le flux vidéo (30 frames par seconde).
Frame Stack	Pile FIFO (First In First Out) stockant les frames horodatées pour l'affichage différé.
FPS (Frames Per Second)	Nombre d'images affichées par seconde. L'application cible 30 fps pour un rendu fluide.
getUserMedia	API Web permettant d'accéder aux périphériques média (caméra, micro) du navigateur.
ImageData	Objet JavaScript contenant les données pixel d'une image sous forme de tableau (RGBA).
Loader	Indicateur visuel (spinner + compteur) affiché pendant la phase de synchronisation.
MediaStream	Flux continu de données vidéo et/ou audio provenant d'une source (caméra).
Miroir / Mirror	Inversion horizontale de l'image (effet symétrique) pour une perception naturelle.
Règle des tiers	Principe de composition visuelle divisant l'image en 9 zones égales (grille 3x3).
Splash Screen	Écran d'accueil affiché au démarrage, nécessitant une interaction utilisateur pour lancer l'app.
Timestamp	Horodatage précis d'un événement, exprimé en millisecondes depuis le 1er janvier 1970 (epoch).

PARTIE 2 : SPÉCIFICATIONS FONCTIONNELLES DÉTAILLÉES

2.1 Description des Fonctionnalités

2.1.1 F01 - Initialisation et Autorisation Caméra

Description : Au chargement de la page, l'application tente d'accéder à la caméra par défaut du système. Si plusieurs caméras sont disponibles, la première détectée est sélectionnée. L'utilisateur doit explicitement autoriser l'accès via la boîte de dialogue native du navigateur.

Séquence technique :

27. Appel à `navigator.mediaDevices.enumerateDevices()` pour lister les périphériques
28. Filtrage des devices de type `videoinput`
29. Requête getUserMedia avec contraintes
30. Gestion des erreurs possibles
31. Connexion du MediaStream à l'élément `<video>` caché
32. Affichage du splash screen avec bouton "CLIQUEZ POUR LANCER"

> ⚠ **Gestion des erreurs critiques :**
> NotFoundError : Aucune caméra → Message "Aucune caméra détectée"
> NotAllowedError : Permission refusée → Message "Veuillez autoriser l'accès"
> NotReadableError : Caméra occupée → Message "Caméra déjà utilisée par une autre application"

2.1.2 F02 - Activation Plein Écran

Description : L'utilisateur clique sur le splash screen, déclenchant le passage en mode plein écran et le démarrage effectif du pipeline vidéo.

Comportement :

- **Desktop (Chrome/Edge/Firefox) :** Appel à `document.documentElement.requestFullscreen()` → Passage immédiat
- **Safari macOS :** Appel à `webkitRequestFullscreen()` → Nécessite interaction utilisateur
- **iOS Safari :** API non supportée → L'application fonctionne en mode fenêtré

> 💡 **Stratégie d'adaptation :**
> Si requestFullscreen() échoue ou n'existe pas, l'application continue en mode fenêtre maximisée (canvas prend 100% viewport). Aucune erreur n'est affichée.

2.1.3 F03 - Phase de Buffering

Description : Après activation, une phase de synchronisation remplit le buffer temporel avant le premier affichage. Un loader avec compteur informe l'utilisateur.

Algorithme détaillé :

```
config.delayMs = 7000
frameStack = []
isBuffering = true

function loop() {
   now = Date.now()
   ctx.drawImage(videoElement, 0, 0)
   imageData = ctx.getImageData(0, 0, width, height)
   frameStack.push({data: imageData, time: now})
   targetTime = now - config.delayMs
   targetFrame = frameStack.find(f => f.time <= targetTime)
   if (targetFrame) {
      outputCtx.putImageData(targetFrame.data, 0, 0)
      isBuffering = false
   }
}
```

Durée de buffering : Proportionnelle au délai configuré (3s → 3s de buffering, 11s → 11s de buffering)

2.1.5 F05 - Grille de Visée (Règle des Tiers)

Description : Overlay semi-transparent traçant 4 lignes (2 horizontales, 2 verticales) divisant l'écran en 9 zones égales.

Palette de couleurs disponibles :

Couleur	Code RGB	Usage recommandé
Blanc	rgb(255, 255, 255)	Fond sombre, éclairage faible
Jaune	rgb(255, 255, 0)	Contraste universel (défaut)
Cyan	rgb(0, 255, 255)	Fond chaud (orange/rouge)
Magenta	rgb(255, 0, 255)	Fond vert
Vert lime	rgb(50, 255, 50)	Intérieur, éclairage neutre
Orange	rgb(255, 165, 0)	Fond bleu/froid
Rouge	rgb(255, 50, 50)	Maximum contraste, attention

2.1.6 F06 - Menu de Configuration

Description : Fenêtre modale accessible en cliquant n'importe où sur le canvas (hors période de buffering). Permet de modifier : caméra source, délai temporel, couleur de grille.

Actions et Conséquences :

Action	Effet immédiat	Rebuffering requis ?
Changement de caméra	Arrêt du flux actuel, redémarrage avec nouveau deviceId	✅ Oui (durée = délai configuré)
Changement de délai	Vidage de frameStack, mise à jour de config.delayMs	✅ Oui (durée = nouveau délai)
Changement couleur grille	Mise à jour de config.gridColor	❌ Non (changement instantané)

3. Annexes Techniques

3.1 Spécifications des API Web

3.1.1 getUserMedia - Contraintes Vidéo

```
const constraints = {
   audio: false,
   video: {
      width: { ideal: 1280 },
      height: { ideal: 720 },
      frameRate: { ideal: 30, max: 30 },
      facingMode: "user"
   }
}
```

3.2 Performances et Benchmarks

3.2.1 Métriques de Performance

Mesure	Méthode	Valeur cible

FPS effectif	Compteur de frames sur 1 seconde	28-30 fps stable
Latence système	Timestamp capture → timestamp affichage - délai configuré	< 50 ms
Taille buffer	frameStack.length	délai × FPS ± 10%
Usage CPU	Performance Monitor navigateur	20-30% d'un cœur
Usage RAM	Task Manager / Activity Monitor	280-500 MB selon délai

Profil de Consommation : Configuration test : PC i5-8250U, 8GB RAM, Webcam 720p Résultats moyens (délai 7s) : • CPU : 20-25% d'un cœur • RAM : 280-320 MB • GPU : ~5% (effet miroir) • FPS maintenu : 30 stable

3.3 Gestion des Erreurs

3.3.1 Codes d'Erreur getUserMedia

Code	Signification	Action Utilisateur
NotAllowedError	Permission refusée	Autoriser caméra dans paramètres navigateur
NotFoundError	Aucune caméra détectée	Connecter une webcam
NotReadableError	Caméra déjà utilisée	Fermer autres applications utilisant la caméra
OverconstrainedError	Contraintes impossibles	Caméra ne supporte pas 720p → Réessai auto en 480p
SecurityError	Contexte non sécurisé	Utiliser HTTPS ou localhost

3.5 Tests et Validation

3.5.1 Scénarios de Test

Test	Procédure	Résultat attendu
T01 - Premier lancement	Ouvrir fichier, autoriser caméra, cliquer splash	Image miroir après 7s
T02 - Changement délai	Menu → 5s → Attendre	Nouveau délai actif, compteur à 5s
T03 - Changement caméra	Menu → Autre caméra → Attendre	Nouvelle source affichée
T04 - Refus permission	Bloquer caméra dans navigateur	Message d'erreur explicite
T05 - Plein écran	Clic splash	Passage fullscreen (sauf iOS)
T06 - Quitter (Ctrl+Q)	Appuyer Ctrl+Q	Écran "Application fermée"
T07 - Stabilité 30 min	Laisser tourner 30 minutes	Pas de fuite mémoire, FPS stable

3.5.2 Validation Multiplateforme

Plateforme	Navigateur	Statut	Notes
Windows 10/11	Chrome 120+	✅ 100%	Référence développement
Windows 10/11	Edge 120+	✅ 100%	Identique à Chrome
Windows 10/11	Firefox 121+	✅ 95%	Légère différence rendu grille
macOS Sonoma	Safari 17+	✅ 95%	Fullscreen nécessite interaction

macOS	Chrome 120+	✅ 100%	-
Ubuntu 22.04	Firefox 121+	✅ 100%	-
Android 12+	Chrome	✅ 90%	Fullscreen OK, performances variables
iOS 17+	Safari	⚠ 80%	Pas de fullscreen iPhone

3.6 Évolutions Futures

3.6.1 Améliorations Envisagées (V4.0)

Fonctionnalités potentielles :

- 📸 Capture instantanée (screenshot) d'une frame pour comparaison
- 🎨 Couleur grille personnalisable depuis le menu
- 📊 Mode comparaison côte-à-côte (2 délais simultanés)
- 🎯 Superposition de repères personnalisés (lignes d'alignement)
- 💾 Enregistrement local optionnel (dans IndexedDB)
- 🌐 Mode multi-caméras (affichage 2 angles simultanés)

3.6.2 Optimisations Techniques

- Passage à WebGL pour rendu GPU-accéléré
- Worker thread pour buffering (libérer thread principal)
- Compression frames (JPEG dans buffer au lieu de raw ImageData)
- Support 60fps pour caméras haute performance