MindAR.js Flow Diagrams

Complete System Architecture and Data Flow

Table of Contents

---

1. High-Level System Architecture

```
graph TB
    subgraph "User Input"
        A[Marker Images] --> B[Compiler]
        V[Video Stream] --> C[Controller]
    end

    subgraph "Compilation (Offline)"
        B --> D[.mind File]
    end

    subgraph "Runtime (Online)"
        D --> C
        C --> E[Detector]
        E --> F[Matcher]
        F --> G[Estimator]
        G --> H[Tracker]
        H --> G
        G --> I[3D Pose Output]
    end

    subgraph "Rendering"
        I --> J[Three.js / A-Frame]
        J --> K[AR Experience]
    end

    style A fill:#e1f5ff
    style V fill:#e1f5ff
    style D fill:#ffe1e1
```

```
    style I fill:#e1ffe1
    style K fill:#ffe1ff
```

---

2. Compilation Flow (Offline)

```
flowchart TD
    Start([User Provides Image]) --> Canvas[Create Canvas Element]
    Canvas --> Gray[Convert to Grayscale<br/>RGB → Average]

    Gray --> Pyramid[Build Image Pyramid<br/>buildImageList]
    Pyramid --> Scales{For Each Scale}

    Scales --> Detect[Detector.detect]

    subgraph "Detector Process"
        Detect --> Gauss[Build Gaussian Pyramid<br/>5 octaves max]
        Gauss --> DoG[Compute DoG Images<br/>img2 - img1]
        DoG --> Extrema[Find Extrema<br/>Local max/min in 3x3x3]
        Extrema --> Prune[Prune to 500 Features<br/>10x10 buckets, 5 per bucket]
        Prune --> Localize[Sub-pixel Localization<br/>Quadratic fit]
        Localize --> Orient[Compute Orientations<br/>36-bin histogram]
        Orient --> Freak[Extract FREAK Descriptors<br/>37 points, 666 bits]
    end

    Freak --> Features[Feature Points Array]
    Features --> Split[Split Maxima/Minima]
    Split --> KD[Build K-D Trees<br/>hierarchicalClustering]

    KD --> NextScale{More Scales?}
    NextScale -->|Yes| Scales
    NextScale -->|No| Matching[Matching Data Complete]

    Matching --> TrackPyr[Build Tracking Pyramid<br/>2 scales: 256px, 128px]
    TrackPyr --> Worker[Spawn Worker Thread]
    Worker --> TrackExtract[Extract Tracking Features]
    TrackExtract --> Tracking[Tracking Data Complete]

    Tracking --> Combine[Combine Data]
    Combine --> Export[msgpack.encode]
    Export --> Mind[Save .mind File]
    Mind --> End([Compilation Done])

    style Start fill:#e1f5ff
    style Mind fill:#ffe1e1
    style End fill:#e1ffe1
```

---

3. Runtime Detection Flow (Online)

```
flowchart TD
    Start([Video Frame]) --> Input[InputLoader<br/>WebGL Grayscale]

    Input --> Crop[CropDetector<br/>Find Moving Regions]
    Crop --> Detect[Detector.detect<br/>Extract Features]

    Detect --> Worker[Send to Worker Thread]

    subgraph "Worker Thread"
        Worker --> Match[Matcher.matchDetection<br/>Try all keyframe scales]

        subgraph "For Each Keyframe"
            Match --> KDSearch[K-D Tree Search<br/>Find candidates]
            KDSearch --> Hamming[Hamming Distance<br/>Compare descriptors]
            Hamming --> Lowe[Lowe's Ratio Test<br/>Filter ambiguous]
            Lowe --> Hough[Hough Voting<br/>Geometric consistency]
            Hough --> RANSAC[RANSAC Homography<br/>Robust estimation]
            RANSAC --> Inlier[Inlier Filtering<br/>< 3px error]
            Inlier --> Pass2[Second Pass Matching<br/>Homography-guided]
        end

        Pass2 --> BestFrame[Select Best Keyframe<br/>Most matches]
        BestFrame --> Coords[Convert to 3D Coords<br/>Screen + World]

        Coords --> EstWorker[Estimator.estimate<br/>DLT Algorithm]
        EstWorker --> Pose[4x4 ModelView Matrix]
    end

    Pose --> MainThread[Return to Main Thread]
    MainThread --> Filter[One-Euro Filter<br/>Smooth pose]

    Filter --> Track{Tracking Mode?}
    Track -->|No| Output
    Track -->|Yes| Template[Template Matching<br/>Faster than detection]
    Template --> Refine[ICP Refinement<br/>Minimize error]
    Refine --> Output[3D Pose Output]

    Output --> NextFrame{Next Frame?}
    NextFrame -->|Yes| Input
    NextFrame -->|No| End([Stop])

    style Start fill:#e1f5ff
    style Output fill:#e1ffe1
    style End fill:#ffe1e1
```

---

4. Detector Pipeline (Feature Extraction)

```
flowchart TD
    Input([Grayscale Image Tensor]) --> Init[Initialize Detector<br/>Calculate
```

```
numOctaves]

    subgraph "Step 1: Gaussian Pyramid"
        Init --> Oct0[Octave 0: Full Resolution]
        Oct0 --> Blur0[Apply Binomial Filter<br/>1,4,6,4,1 kernel]
        Blur0 --> Oct1[Octave 1: Downsample 2x]
        Oct1 --> Blur1[Apply Binomial Filter]
        Blur1 --> Oct2[Octave 2: Downsample 4x]
        Oct2 --> More[... up to 5 octaves]
    end

    More --> DoGStart[DoG Pyramid Generation]

    subgraph "Step 2: DoG Images"
        DoGStart --> Sub0[DoG0 = Blur2 - Blur1]
        Sub0 --> Sub1[DoG1 = Blur2 - Blur1]
        Sub1 --> SubN[DoG2, DoG3, ...]
    end

    SubN --> ExtremaStart[Extrema Detection]

    subgraph "Step 3: Find Extrema"
        ExtremaStart --> Compare[For each pixel in DoG1]
        Compare --> Check1{Value > 3.0?}
        Check1 -->|No| Skip1[Skip]
        Check1 -->|Yes| Check2{Local max/min<br/>vs 26 neighbors?}
        Check2 -->|No| Skip1
        Check2 -->|Yes| Check3{Edge test<br/>Hessian ratio?}
        Check3 -->|Fail| Skip1
        Check3 -->|Pass| Keep1[Keep as extrema]
    end

    Keep1 --> PruneStart[Pruning Process]

    subgraph "Step 4: Prune Extrema"
        PruneStart --> Divide[Divide into 10x10 grid<br/>100 buckets]
        Divide --> Sort[Sort by strength<br/>in each bucket]
        Sort --> Top[Keep top 5 per bucket]
        Top --> Result1[Max 500 features]
    end

    Result1 --> LocalStart[Localization]

    subgraph "Step 5: Sub-pixel Localization"
        LocalStart --> Sample[Sample 3x3 around extrema]
        Sample --> Quad[Fit quadratic surface]
        Quad --> Peak[Find peak position]
        Peak --> Adjust[Adjust x,y coordinates]
    end

    Adjust --> OrientStart[Orientation Computation]
```

```
    subgraph "Step 6: Orientation"
        OrientStart --> Region[Sample circular region]
        Region --> Grad[Compute gradients]
        Grad --> Hist[Build 36-bin histogram<br/>Weighted by magnitude]
        Hist --> Smooth[Gaussian smoothing<br/>5 iterations]
        Smooth --> PeakFind[Find peak bin]
        PeakFind --> Angle[Compute angle −π to π]
    end

    Angle --> FreakStart[FREAK Sampling]

    subgraph "Step 7: FREAK Descriptors"
        FreakStart --> Points[Sample 37 FREAK points<br/>Rotated by angle]
        Points --> Interp[Bilinear interpolation]
        Interp --> Values[37 intensity values]
        Values --> Compare2[Compare all pairs<br/>666 comparisons]
        Compare2 --> Binary[666 binary bits]
        Binary --> Pack[Pack into 167 integers]
    end

    Pack --> Output([Feature Points Array<br/>x, y, scale, angle, descriptors])

    style Input fill:#e1f5ff
    style Output fill:#e1ffe1
    style Result1 fill:#ffe1e1
```

---

5. Matching Pipeline

```
flowchart TD
    Start([Query Features + Keyframe]) --> Loop{For each<br/>query point}

    Loop --> Type{Maxima or<br/>Minima?}
    Type -->|Maxima| KDMax[K-D Tree: maximaCluster]
    Type -->|Minima| KDMin[K-D Tree: minimaCluster]

    subgraph "K-D Tree Search"
        KDMax --> Traverse[Traverse tree]
        KDMin --> Traverse
        Traverse --> Children[Compare to children]
        Children --> Best[Follow best path]
        Best --> Backtrack[Backtrack up to 8 nodes]
        Backtrack --> Candidates[Candidate keypoints]
    end

    Candidates --> HammLoop{For each<br/>candidate}
    HammLoop --> Hamm[Compute Hamming Distance<br/>XOR + popcount]
    Hamm --> Track[Track best & 2nd-best]
    Track --> HammNext{More<br/>candidates?}
    HammNext -->|Yes| HammLoop
    HammNext -->|No| Ratio{Best/2nd < 0.7?}
```

```
Ratio -->|No| Reject1[Reject match]
Ratio -->|Yes| Accept1[Accept match]

Accept1 --> LoopNext{More<br/>queries?}
Reject1 --> LoopNext
LoopNext -->|Yes| Loop
LoopNext -->|No| CheckCount{Matches >= 6?}

CheckCount -->|No| Failed([No Match])
CheckCount -->|Yes| Hough[Hough Voting]

subgraph "Hough Transform"
    Hough --> Bins[Create bins:<br/>scale, rotation, position]
    Bins --> Vote[Each match votes]
    Vote --> Peak[Find peak bin]
    Peak --> HoughMatches[Filtered matches]
end

HoughMatches --> RANSAC[RANSAC Homography]

subgraph "RANSAC Process"
    RANSAC --> Sample[Sample 4 random matches]
    Sample --> Compute[Compute homography H]
    Compute --> Test[Test all matches]
    Test --> Count[Count inliers < 3px]
    Count --> Iter{More<br/>iterations?}
    Iter -->|Yes| Sample
    Iter -->|No| Best2[Select best H]
end

Best2 --> Inliers[Filter inliers]
Inliers --> Check2{Inliers >= 6?}
Check2 -->|No| Failed

Check2 -->|Yes| Pass2Start[Second Pass Matching]

subgraph "Homography-Guided Pass"
    Pass2Start --> Invert[Invert homography H⁻¹]
    Invert --> Warp[Warp query points to keyframe space]
    Warp --> Search[Search within 10px radius]
    Search --> Hamm2[Hamming distance matching]
    Hamm2 --> Ratio2[Lowe's ratio test]
end

Ratio2 --> Hough2[Hough voting again]
Hough2 --> RANSAC2[RANSAC again]
RANSAC2 --> Final[Final inlier matches]

Final --> Success([Homography H + Matches])

style Start fill:#e1f5ff
```

```
    style Success fill:#e1ffe1
    style Failed fill:#ffe1e1
```

---

6. Pose Estimation Pipeline

```
flowchart TD
    Start([2D-3D Correspondences]) --> Screen[Screen Coords: pixel x,y]
    Start --> World[World Coords: normalized X,Y,Z]

    Screen --> DLT[DLT Algorithm]
    World --> DLT

    subgraph "Direct Linear Transform"
        DLT --> System[Build linear system<br/>Ax = 0]
        System --> SVD[Singular Value Decomposition]
        SVD --> Homo[Extract Homography H<br/>3x3 matrix]
    end

    Homo --> Decomp[Homography Decomposition]

    subgraph "Decompose H into R,t"
        Decomp --> Normalize[Normalize H]
        Normalize --> Extract[Extract rotation columns]
        Extract --> Ortho[Orthogonalize rotation<br/>SVD on R]
        Ortho --> Trans[Extract translation t]
        Trans --> Build[Build 4x4 matrix]
    end

    Build --> Initial[Initial ModelView Matrix]

    Initial --> Refine{Refinement<br/>needed?}
    Refine -->|No| Output
    Refine -->|Yes| ICP[ICP Refinement]

    subgraph "Iterative Closest Point"
        ICP --> Project[Project 3D world points<br/>using current pose]
        Project --> Error[Compute reprojection errors<br/>2D screen vs projected]
        Error --> Jacobian[Compute Jacobian matrix<br/>∂error/∂pose]
        Jacobian --> Update[Levenberg-Marquardt update<br/>Minimize squared error]
        Update --> NewPose[Updated pose]
        NewPose --> Converge{Converged?}
        Converge -->|No| Project
        Converge -->|Yes| Refined[Refined ModelView]
    end

    Refined --> Output([6-DOF Camera Pose<br/>4x4 matrix])

    style Start fill:#e1f5ff
```

```
    style Output fill:#e1ffe1
    style Initial fill:#fff4e1
```

---

7. Controller Main Loop

```
flowchart TD
    Start([Controller.start]) --> LoadMind[Load .mind file<br/>importData]
    LoadMind --> InitVideo[Initialize video stream<br/>getUserMedia]

    InitVideo --> CreateWorker[Create Worker Thread<br/>ControllerWorker]
    CreateWorker --> SendData[Send compiled data to worker]

    SendData --> Loop[Main Loop: processVideo]

    subgraph "Frame Processing"
        Loop --> GetFrame[Get video frame]
        GetFrame --> InputLoad[InputLoader<br/>WebGL grayscale conversion]
        InputLoad --> Crop[CropDetector<br/>Detect movement]

        Crop --> Mode{Current Mode?}

        Mode -->|Detection| DetectFull[Full Detection<br/>Detector.detect]
        Mode -->|Tracking| TrackTemplate[Template Matching<br/>Tracker.track]

        DetectFull --> SendWorker[Post to Worker:<br/>featurePoints]
        TrackTemplate --> SendWorker

        SendWorker --> Wait[Wait for worker response]

        Wait --> WorkerResp{Worker Result?}
        WorkerResp -->|No match| Loop
        WorkerResp -->|Match found| Filter[One-Euro Filter<br/>Smooth pose]

        Filter --> ModeSwitch{Match quality?}
        ModeSwitch -->|Good| SetTrack[Switch to Tracking mode]
        ModeSwitch -->|Lost| SetDetect[Switch to Detection mode]

        SetTrack --> Emit[Emit 'targetFound' event]
        SetDetect --> Emit2[Emit 'targetLost' event]

        Emit --> UpdateAR[Update AR scene<br/>Three.js/A-Frame]
        Emit2 --> UpdateAR
    end

    UpdateAR --> RAF{requestAnimationFrame}
    RAF -->|Continue| Loop
    RAF -->|Stop| Cleanup[Cleanup resources]

    Cleanup --> End([Controller.stop])
```

```
    style Start fill:#e1f5ff
    style End fill:#ffe1e1
    style UpdateAR fill:#e1ffe1
```

---

8. Worker Thread Communication

```
sequenceDiagram
    participant Main as Main Thread<br/>(Controller)
    participant Worker as Worker Thread<br/>(ControllerWorker)
    participant Match as Matcher
    participant Est as Estimator

    Note over Main,Worker: Initialization Phase
    Main->>Worker: postMessage({type: 'init', data: compiledData})
    Worker->>Worker: Load keyframes
    Worker->>Main: {type: 'initDone'}

    Note over Main,Worker: Detection Loop
    Main->>Main: Detector.detect(frame)
    Main->>Worker: {type: 'match', featurePoints}

    Worker->>Match: matchDetection(keyframes, featurePoints)

    loop For each keyframe scale
        Match->>Match: K-D tree search
        Match->>Match: Hamming distance
        Match->>Match: Hough voting
        Match->>Match: RANSAC homography
    end

    Match-->>Worker: {screenCoords, worldCoords, keyframeIndex}

    alt Match found
        Worker->>Est: estimate({screenCoords, worldCoords})
        Est-->>Worker: modelViewTransform
        Worker->>Main: {type: 'found', pose: modelViewTransform}
        Main->>Main: One-Euro filter
        Main->>Main: Update AR scene
    else No match
        Worker->>Main: {type: 'notFound'}
        Main->>Main: Continue detection mode
    end

    Note over Main,Worker: Tracking Loop (after match found)
    Main->>Main: Tracker.track(frame, lastPose)
    Main->>Worker: {type: 'trackMatch', featurePoints}
    Worker->>Match: matchDetection (fewer features)
    Match-->>Worker: {screenCoords, worldCoords}
    Worker->>Est: refineEstimate({initialPose, coords})
```

```
    Est-->>Worker: refinedModelViewTransform
    Worker->>Main: {type: 'found', pose: refinedPose}
```

---

9. Data Structure Flow

```
flowchart LR
    subgraph "Input"
        A[Image File<br/>PNG/JPG]
        B[Video Frame<br/>ImageData]
    end

    subgraph "Preprocessing"
        C[Grayscale Array<br/>Uint8Array<br/>width × height]
    end

    subgraph "Image Pyramid"
        D[Scale List<br/>[1.0, 0.79, 0.63, ...]]
        E[Image List<br/>[{data, w, h, scale}, ...]]
    end

    subgraph "Detection"
        F[Pyramid Tensors<br/>[[img1, img2], ...]]
        G[DoG Tensors<br/>[diff0, diff1, ...]]
        H[Extrema Tensors<br/>[extrema0, extrema1, ...]]
    end

    subgraph "Pruning"
        I[Pruned List<br/>[[score, oct, y, x], ...]]
        J[Localized Tensor<br/>refined positions]
    end

    subgraph "Descriptors"
        K[Orientation Tensor<br/>angles −π to π]
        L[FREAK Tensor<br/>37 × features]
        M[Binary Tensor<br/>666 bits × features]
    end

    subgraph "Feature Points"
        N["Feature Array<br/>[{x, y, scale, angle,<br/>descriptors[]}, ...]"]
    end

    subgraph "Clustering"
        O[Maxima Array<br/>bright features]
        P[Minima Array<br/>dark features]
        Q[K-D Tree Maxima<br/>{rootNode, ...}]
        R[K-D Tree Minima<br/>{rootNode, ...}]
    end

    subgraph "Keyframe"
        S["Keyframe Object<br/>{maximaPoints,<br/>minimaPoints,<br/>maximaCluster,
```

```
<br/>minimaCluster,<br/>width, height, scale}"]
    end

    subgraph "Compiled Output"
        T[".mind File<br/>MessagePack binary<br/>[{matchingData,<br/>trackingData},
...]"]
    end

    subgraph "Runtime Matching"
        U[Match Array<br/>[{querypoint, keypoint}, ...]]
        V[Homography H<br/>3×3 matrix]
        W[Inlier Matches<br/>filtered < 3px]
    end

    subgraph "Pose"
        X["2D–3D Pairs<br/>{screenCoords:[{x,y}],<br/>worldCoords:[{X,Y,Z}]}"]
        Y[ModelView Matrix<br/>4×4 transform]
    end

    A --> C
    B --> C
    C --> D
    D --> E
    E --> F
    F --> G
    G --> H
    H --> I
    I --> J
    J --> K
    K --> L
    L --> M
    M --> N
    N --> O
    N --> P
    O --> Q
    P --> R
    Q --> S
    R --> S
    S --> T

    N --> U
    U --> V
    V --> W
    W --> X
    X --> Y

    style A fill:#e1f5ff
    style B fill:#e1f5ff
    style T fill:#ffe1e1
    style Y fill:#e1ffe1
```

---

10. Class Hierarchy

```
classDiagram
    class CompilerBase {
        +data
        +compileImageTargets(images, callback)
        +exportData()
        +importData(buffer)
        #createProcessCanvas(img)*
        #compileTrack(options)*
    }

    class Compiler {
        +createProcessCanvas(img)
        +compileTrack(options)
    }

    class Detector {
        -width
        -height
        -numOctaves
        -tensorCaches
        -kernelCaches
        +detect(inputImageT)
        -_applyFilter(image)
        -_downsampleBilinear(image)
        -_buildExtremas(img0, img1, img2)
        -_applyPrune(extremas)
        -_computeLocalization(list, dogs)
        -_computeOrientationHistograms(extremas, pyramids)
        -_computeExtremaFreak(pyramids, extremas, angles)
        -_computeFreakDescriptors(freaks)
    }

    class Matcher {
        -queryWidth
        -queryHeight
        -debugMode
        +matchDetection(keyframes, featurePoints)
    }

    class Estimator {
        -projectionTransform
        +estimate(screenCoords, worldCoords)
        +refineEstimate(initialPose, coords)
    }

    class Tracker {
        -projectionTransform
        -width
```

```
        -height
        +track(imageList, lastPose)
    }

    class Controller {
        -worker
        -detector
        -matcher
        -estimator
        -tracker
        -inputLoader
        -filter
        +start()
        +stop()
        -processVideo()
    }

    class InputLoader {
        -width
        -height
        -gl
        +loadImage(imageData)
    }

    class OneEuroFilter {
        -config
        +filter(value, timestamp)
    }

    CompilerBase <|-- Compiler : extends
    Controller --> Detector : uses
    Controller --> Matcher : uses (via worker)
    Controller --> Estimator : uses (via worker)
    Controller --> Tracker : uses
    Controller --> InputLoader : uses
    Controller --> OneEuroFilter : uses
    Compiler --> Detector : uses
    Matcher --> Detector : uses features from

    note for CompilerBase "* = abstract method<br/>subclass must implement"
```

---

11. File Organization Flow

```
graph TD
    subgraph "Entry Points"
        A[src/image-target/index.js]
    end

    subgraph "Compilation"
        B[compiler.js]
```

```
    C[compiler-base.js]
    D[compiler.worker.js]
    E[offline-compiler.js]
end

subgraph "Detection & Features"
    F[detector/detector.js]
    G[detector/freak.js]
    H[detector/kernels/webgl/]
    I[detector/kernels/cpu/]
    J[image-list.js]
end

subgraph "Matching"
    K[matching/matcher.js]
    L[matching/matching.js]
    M[matching/hierarchical-clustering.js]
    N[matching/hamming-distance.js]
    O[matching/hough.js]
    P[matching/ransacHomography.js]
end

subgraph "Pose Estimation"
    Q[estimation/estimator.js]
    R[estimation/estimate.js]
    S[estimation/refine-estimate.js]
    T[estimation/utils.js]
end

subgraph "Tracking"
    U[tracker/tracker.js]
    V[tracker/extract.js]
    W[tracker/extract-utils.js]
end

subgraph "Runtime Control"
    X[controller.js]
    Y[controller.worker.js]
    Z[input-loader.js]
    AA[crop-detector.js]
end

subgraph "Integration"
    AB[three.js]
    AC[aframe.js]
end

A --> B
A --> X
B --> C
C --> D
C --> F
```

```
    C --> J

    F --> G
    F --> H
    F --> I

    X --> Y
    X --> Z
    X --> AA
    X --> F
    X --> U

    Y --> K
    Y --> Q

    K --> L
    L --> M
    L --> N
    L --> O
    L --> P

    Q --> R
    Q --> S
    Q --> T

    U --> V
    U --> W

    X --> AB
    X --> AC

    style A fill:#e1f5ff
    style AB fill:#e1ffe1
    style AC fill:#e1ffe1
```

---

12. Algorithm Complexity Overview

```
graph TB
    subgraph "Compilation - O(N)"
        A[Load Image<br/>O1: Single pass]
        A --> B[Build Pyramid<br/>O2: Log scales]
        B --> C[Detect Features<br/>O3: W×H per scale]
        C --> D[Prune to 500<br/>O4: Linear scan]
        D --> E[Build K-D Tree<br/>O5: N log N]
    end

    subgraph "Runtime Detection - O(M)"
        F[Detect Features<br/>O6: W×H]
        F --> G[K-D Tree Search<br/>O7: M log N]
        G --> H[Hamming Distance<br/>O8: M×k×167 XOR]
```

```
        H --> I[Hough Voting<br/>O9: M bins]
        I --> J[RANSAC<br/>O10: Iterations×4]
        J --> K[DLT<br/>O11: SVD 3×3]
    end

    subgraph "Runtime Tracking - O(M)"
        L[Template Track<br/>O12: Small region]
        L --> M[Match Few Points<br/>O13: M×k]
        M --> N[ICP Refine<br/>O14: Iterations×M]
    end

    E --> G
    K --> L

    style A fill:#e1f5ff
    style F fill:#fff4e1
    style L fill:#e1ffe1
```

Legend:

- N = Number of compiled features (~500)
- M = Number of query features (~100-500)
- k = K-D tree candidates (~10-50)
- W, H = Image width, height

---

13. Memory Layout

```
graph LR
    subgraph "GPU Memory (TensorFlow.js)"
        A[Input Tensor<br/>W×H float32]
        A --> B[Pyramid Tensors<br/>~5 octaves × 2]
        B --> C[DoG Tensors<br/>~5 octaves]
        C --> D[Feature Tensors<br/>500×167 int32]
    end

    subgraph "CPU Memory (JavaScript)"
        E[Feature Array<br/>~80KB per scale]
        F[K-D Tree<br/>~200KB per scale]
        G[Compiled Data<br/>~2-10MB per target]
    end

    subgraph "Worker Memory"
        H[Keyframes Copy<br/>Same as compiled]
        I[Match State<br/>~100KB]
    end

    D -.Transfer.-> E
    E --> F
    F --> G
    G -.postMessage.-> H
```

```
    style A fill:#e1f5ff
    style G fill:#ffe1e1
    style H fill:#fff4e1
```

---

14. Threading Model

```
graph TB
    subgraph "Main Thread"
        A[Video Frame Capture]
        A --> B[InputLoader WebGL]
        B --> C[Detector TensorFlow.js]
        C --> D{Mode?}
        D -->|Detection| E[Full Feature Extraction]
        D -->|Tracking| F[Template Matching]
        E --> G[Post to Worker]
        F --> G

        H[Worker Response] --> I[One-Euro Filter]
        I --> J[Update Three.js Scene]
        J --> K[Render AR]
    end

    subgraph "Worker Thread"
        G -.Message.-> L[Receive Features]
        L --> M[Matcher]
        M --> N{Match?}
        N -->|Yes| O[Estimator]
        N -->|No| P[No-Op]
        O --> Q[Compute Pose]
        Q -.Message.-> H
        P -.Message.-> H
    end

    subgraph "Compiler Worker (Offline)"
        R[Receive Images] --> S[Extract Tracking Features]
        S --> T[Return Data]
    end

    style A fill:#e1f5ff
    style K fill:#e1ffe1
    style R fill:#fff4e1
```

---

15. State Machine (Controller)

```
stateDiagram-v2
    [*] --> Idle
    Idle --> Loading: start()
```

```
    Loading --> Scanning: .mind loaded

    Scanning --> Scanning: No match found
    Scanning --> Found: Match detected

    Found --> Tracking: Good match quality
    Tracking --> Tracking: Template match success
    Tracking --> Scanning: Match lost

    Found --> Scanning: Match quality poor

    Scanning --> Idle: stop()
    Tracking --> Idle: stop()
    Found --> Idle: stop()

    note right of Idle
        No video processing
        Resources released
    end note

    note right of Loading
        Load .mind file
        Initialize video
        Create workers
    end note

    note right of Scanning
        Full feature detection
        Try match all keyframes
        UI: scanning overlay
    end note

    note right of Found
        Match found
        Emit targetFound event
        Decide: track or scan
    end note

    note right of Tracking
        Template matching
        ICP refinement
        Faster than scanning
    end note
```

---

16. Coordinate System Transformations

```
flowchart TD
    subgraph "World Space (Target)"
        A["3D World Point<br/>(X, Y, 0)<br/>Normalized units"]
    end
```

```
    subgraph "Camera Space"
        B["3D Camera Point<br/>(x, y, z)<br/>Meters"]
    end

    subgraph "Normalized Device Coords"
        C["NDC Point<br/>(-1 to 1, -1 to 1)"]
    end

    subgraph "Screen Space (Video)"
        D["2D Screen Point<br/>(u, v)<br/>Pixels"]
    end

    A -->|ModelView Matrix<br/>4×4| B
    B -->|Projection Matrix<br/>4×4| C
    C -->|Viewport Transform<br/>Scale + Offset| D

    D -.->|Estimator.estimate<br/>Inverse pipeline| A

    style A fill:#e1ffe1
    style D fill:#e1f5ff
```

Transformation Equations:

```
Camera Space: [xc, yc, zc, 1]ᵀ = ModelView × [X, Y, 0, 1]ᵀ

NDC: [xn, yn, zn, w]ᵀ = Projection × [xc, yc, zc, 1]ᵀ
     (xn/w, yn/w) ∈ [-1, 1]

Screen: u = (xn/w + 1) × width/2
        v = (yn/w + 1) × height/2
```

---

17. Performance Optimization Strategy

```
flowchart TD
    Start([Frame Arrives]) --> Check1{Target<br/>tracked?}

    Check1 -->|No| FullDetect[Full Detection<br/>~30ms]
    Check1 -->|Yes| FastTrack[Template Tracking<br/>~10ms]

    FullDetect --> Crop[CropDetector<br/>Skip stable regions]
    Crop --> Detect[Detect Features<br/>GPU accelerated]

    Detect --> Cache{Features<br/>cached?}
    Cache -->|Yes| SkipCompute[Use cached]
    Cache -->|No| Compute[Compute descriptors]

    Compute --> Match[Worker: Match<br/>Parallel processing]
    SkipCompute --> Match
```

```
    FastTrack --> Template[Small region search<br/>~50×50 px]
    Template --> Quick[Quick descriptor match<br/>~10 features]

    Match --> Result1{Found?}
    Quick --> Result2{Found?}

    Result1 -->|Yes| ICP1[ICP Refinement<br/>5 iterations]
    Result1 -->|No| UI1[Continue scanning]

    Result2 -->|Yes| ICP2[ICP Refinement<br/>3 iterations]
    Result2 -->|No| Fallback[Fall back to detection]

    ICP1 --> Filter[One-Euro Filter<br/>Smooth jitter]
    ICP2 --> Filter

    Filter --> Render[Render<br/>~1-2ms]
    Render --> End([Next Frame])

    UI1 --> End
    Fallback --> FullDetect

    style FastTrack fill:#e1ffe1
    style FullDetect fill:#ffe1e1
    style Render fill:#e1f5ff
```

Legend for All Diagrams

Colors:

- **Light Blue** (#e1f5ff): Input/Start
- **Light Red** (#ffe1e1): Intermediate/Storage
- **Light Green** (#e1ffe1): Output/Success
- **Light Yellow** (#fff4e1): Processing/Worker

Shapes:

- **Rectangle**: Process/Function
- **Diamond**: Decision/Branch
- **Rounded Rectangle**: Start/End
- **Parallelogram**: Input/Output
- **Cylinder**: Data Storage

Flow diagrams generated for MindAR.js v1.2.5 Focus: Image Target Recognition System