İSDEMİR'de OPC UA Destekli Görüntü İşleme ile Slab Kenar Eğriliği İzleme

Proje Raporu

Hazırlayan: Azra Barbaros

Birim: Seviye 2 Başmühendisliği

Staj Amiri: Zekeriya Sarıkol

Tarih: 14.08.2025

İçindekiler

İSDEMİR'de OPC UA Destekli Görüntü İşleme ile Slab Kenar Eğriliği İzleme Proje Raporu

1) Özet

İsdemir'de sıvı çelik, sürekli döküm hatlarında slablara dönüştürülür; bu slablar ısıtma fırınlarından çıkarıldıktan sonra kaba ve bitirici hadde standlarında hedeflenen ölçü ve yüzey kalitesiyle haddelenir. Süreç sürekliliği ve kalite güvencesi için slab kenar geometrisinin çevrimiçi izlenmesi kritik bir ihtiyaçtır.

Bu çalışma, slab kenar eğriliklerini gerçek zamanlı olarak "düz", "konkav (içbükey)" ve "konveks (dışbükey)" sınıflarıyla belirleyen bir görüntü işleme–endüstriyel haberleşme çözümü sunar. Mimari, KEPServerEX (OPC UA) ile Python/OpenCV hattını birleştirir: sunucudan alınan komut/zaman bilgisiyle videonun ilgili saniyesine konumlanılır; bu karede tanımlı ROI içinde kenarlar Canny ve morfolojik işlemlerle çıkarılır; sol ve sağ kenardan örneklenen noktalar polinomla modellenerek eğriliğin yönü ve düzlüğü birlikte değerlendirilir ve karar üretilir.

Gerçek sahada tetikleme, bandın belirlenen koordinatlarında slabın kamera görüş açısına girmesiyle yapılacaktır; sahadaki bu davranış, simülasyonda 0–10 saniye aralığında döngüsel bir RAMP sinyaliyle benzetim yoluyla temsil edilmiştir. Üretilen kararlar görsel bindirme (overlay) ve CSV olarak arşivlenir; ayrıca OPC UA düzeyinde istemci–sunucu haberleşmesi kurularak geri besleme amacıyla ResultLeft ve ResultRight etiketlerine (String) yazılarak SCADA/PLC katmanına açılır. Bu sayede entegrasyon yapılmadan da algoritmanın doğruluk, kararlılık ve gerçek zamanlı çalışabilirliği nesnel olarak değerlendirilebilmiştir.

2) Problem Tanımı ve Kapsam

Sürekli dökümden çıkıp haddelemeye girecek slabların yan kenar yüzeylerinde, sıcaklık dağılımı, taşıma ve ön şekillendirme etkileri nedeniyle kenar doğrultusunda eğrilik oluşabilir. Burada odak, kenar çizgisinin düz eksenden saparak konkav (içbükey) ya da konveks (dışbükey) bir profile dönüşmesidir. Bu tür sapmalar, hadde sırasında yön kaçması, kenar kesme miktarında artış ve ölçü tekrarlanabilirliğinde düşüş gibi kalite etkilerine yol açabileceğinden, slab kenarının sürekli ve çevrimiçi izlenmesi önem taşır. Bu bağlamda çalışmanın amacı OPC UA üzerinden sağlanan zaman/komut bilgisi ve OpenCV tabanlı görüntü işleme ile slab kenar eğriliğinin gerçek zamanlı tespitidir.

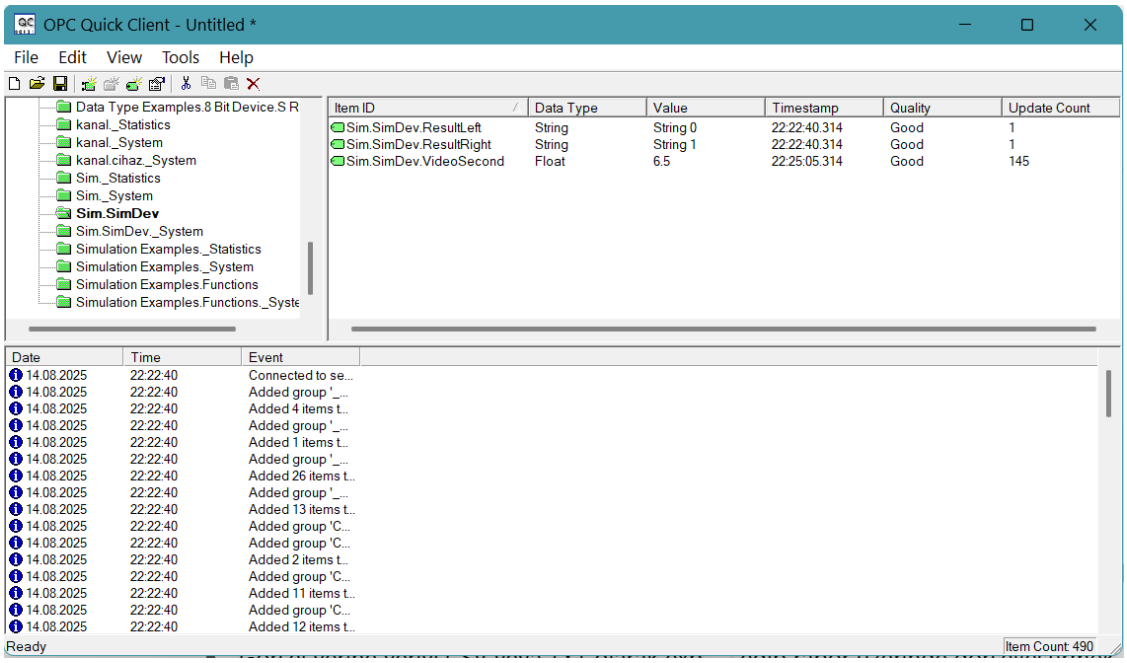
3) Yöntem

3.1. Bileşenler

- **KEPServerEX (OPC UA sunucusu)**

VideoSecond (Float): Python tarafında video kare seçiminde zaman/komut referansı olarak kullanılır; değer 0–10 s aralığında döngüsel akar ve üst sınıra ulaştığında 0'a sarar. Tipik NodeId örneği ns=2;s=Sim.SimDev.VideoSecond olup burada Sim kanal, SimDev cihaz, VideoSecond etiket adıdır. Etiket KEPServerEX'te RAMP(rate_ms, low, high, step) ile üretilir; örneğin RAMP(100, 0.0, 10.0, 1.3) her 100 ms'te 1.3 s artırır (0.0 → 1.3 → 2.6 … → 9.1 → wrap → 0.0). Rate güncelleme periyodu, Low/High alt–üst sınırlar, Step ise her güncellemedeki artıştır. Veri tipi Float, birim saniyedir.

ResultLeft & ResultRight (String): Görüntü işleme sonucunda sol ve sağ kenar için üretilen sınıf kararları bu etiketlere metin olarak yazılır; örnek NodeId'ler ns=2;s=Sim.SimDev.ResultLeft ve ns=2;s=Sim.SimDev.ResultRight'tır. Veri tipi Stringtir; istemci "duz", "icbukey", "disbukey" değerlerini String Variant olarak yazar. Etiketlerin RW (yazılabilir) olması gerekir; KEPServerEX Quick Client üzerinden Synchronous Write ile manuel yazma testi yapmak konfigürasyonu doğrular.



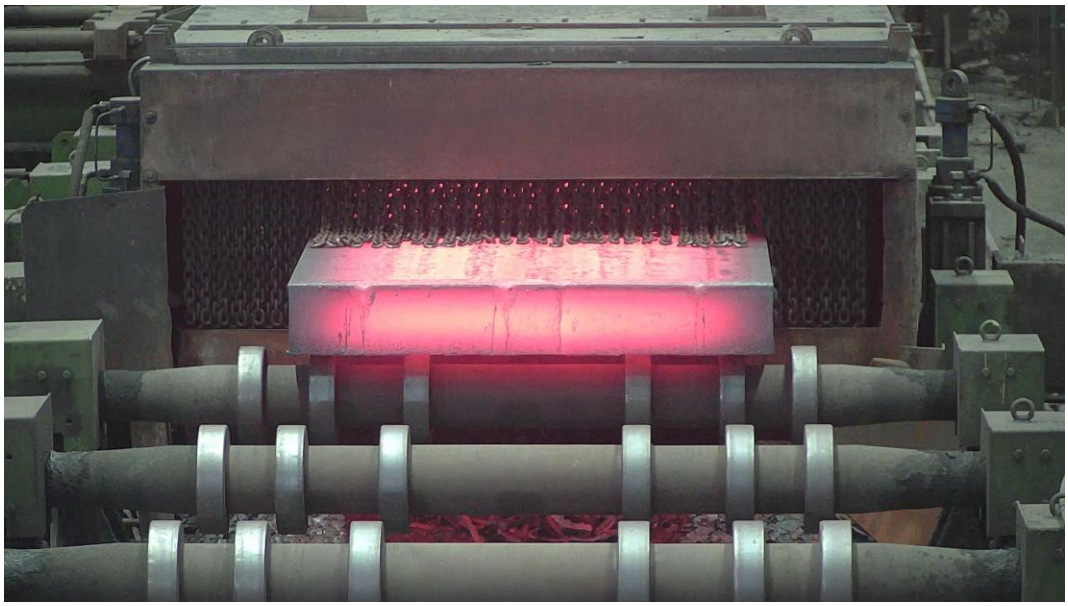
Şekil 1- KEPServerEX Quick Client görünümü. VideoSecond (Float) okuma ve ResultLeft/ResultRight (String) yazma etiketlerinin yapılandırılması.

- **Python uygulaması (asyncua + OpenCV)**

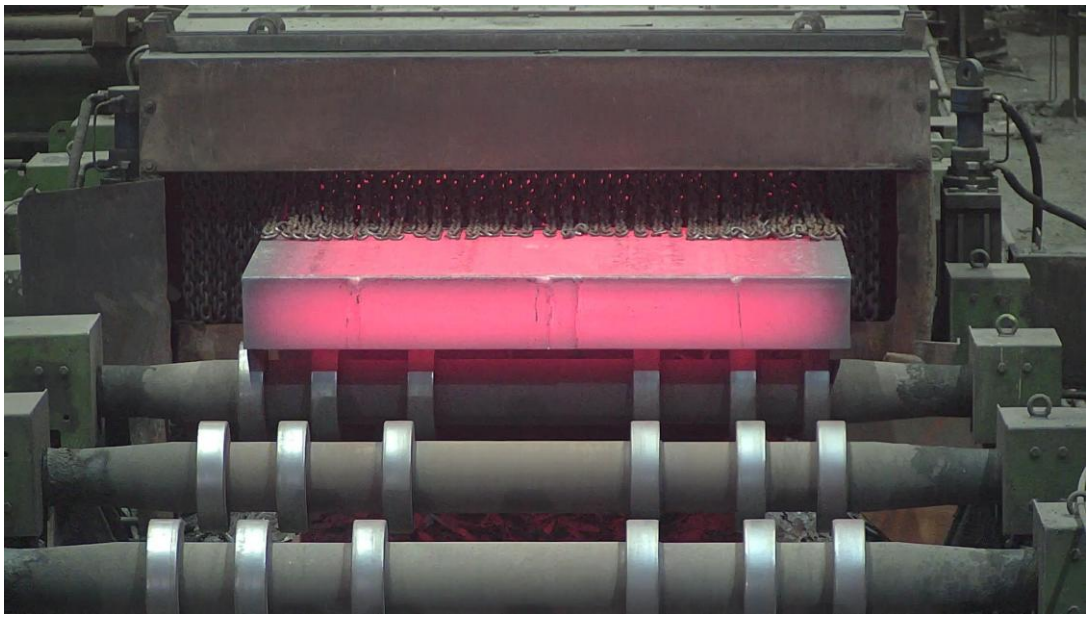
Uygulama, asyncio döngüsü içinde çalışan bir OPC UA istemcisi ve bir görüntü işleme hattından oluşur. İstemci, tanımlanan VideoSecond değerini polling ile okur; her yeni değerde OpenCV videoda o saniyeye seek eder ve ilgili kareyi yakalar. Kare üzerinde ROI → Canny → morfolojik kapama → en büyük bağlı bileşen → 5 satır örnekleme → polyfit + RMS adımları uygulanır; ikinci derece katsayının işareti ile doğruya göre RMS eşiği birlikte değerlendirilerek "düz / içbükey / dışbükey" kararı üretilir. Her tetikte overlay JPG ve CSV kaydı yapılır; 10→0 dönüşleri wrap olarak algılanır ve epoch sayacı artırılarak dosya adlarına/CSV'ye işlenir. (Python kodu Ek'te verilmiştir.)

- **Video veri kaynağı (simülasyon)**

Bu çalışmada canlı kamera akışı bulunmadığından, gerçekte kullanılacak kamera–slab bakı açısını temsil eden iki referans slab fotoğrafından (bkz. Şekil 2 ve Şekil 3) sabit perspektifli 10 saniyelik bir video türetilmiştir. Gerçek sahada planlanan işleyişte görüntü, bandın belirli koordinatlarına gelindiğinde (yani aynı geometrik bakı elde edildiğinde) anlık kare olarak yakalanacaktır. Simülasyonda bu koordinat tetiklemesi, KEPServerEX'teki VideoSecond (0–10 s, döngüsel) etiketiyle zaman tabanlı tetik olarak temsil edilmiştir.



Şekil 2- Referans slab görüntüsü-1. Sabit perspektifte çekilmiş, video üretiminde kullanılan ilk referans kare.



Şekil 3- Referans slab görüntüsü-2. Sabit perspektifte çekilmiş, video üretiminde kullanılan ikinci referans kare.

3.2. Görüntü işleme ve karar verme adımları

Parametre değerleri (ör. alpha, beta, CANNY_MIN, CANNY_MAX, CLOSE_ITER, ROI oranları vb.) çeşitli denemeler ve gözlemler sonucunda optimum performansı sağlayacak şekilde belirlenmiştir.

Parametre	Açıklama	Varsayılan
CANNY_MIN	Canny alt eşik	0
CANNY_MAX	Canny üst eşik	74
CLOSE_ITER	Morfolojik kapama iterasyonu	5
ALPHA, BETA	Kontrast/brightness	3.0, 50
ROI_Y0/Y1	Düşey ROI yüzdeleri	0.47 / 0.58
ROI_X0/X1	Yatay ROI yüzdeleri	0.15 / 0.85
A_THR, RMS_THR	Karar eşikleri	1.25(0.09–0.2 aralığı) / 5.5

Tablo 1- Görüntü işleme parametreleri ve varsayılan değerler.

Görüntü işleme ve karar adımları şu şekilde uygulanmıştır:

i. Ön işleme uygulanır: resize(960×540) → convertScaleAbs(alpha, beta) → gri → GaussianBlur(5×3).
ii. Kenar tespiti yapılır: Canny(CANNY_MIN, CANNY_MAX) → morph close (3×3, iter=CLOSE_ITER).
iii. Maskeleme yapılır: bitwise_not(edges) ile beyaz = nesne olacak şekilde maske oluşturulur.
iv. ROI seçilir: y ∈ [0.47H, 0.58H], x ∈ [0.15W, 0.85W] aralığında bölge belirlenir.
v. Bağlı bileşen işlemi uygulanır: ROI'de en büyük beyaz alan bulunup mask_big içine yerleştirilir.
vi. Örnekleme yapılır: ROI'de 5 yatay satırda aynı y_i konumunda x_left(x_i^L) ve x_right(x_i^R) noktaları çıkarılır. Sol kenar:(y_i, x_i^L), Sağ kenar:(y_i, x_i^R)
vii. Sayısal kararlılık için y_i değerleri 0–1'e normalize edilir:

$$y_i^N = \frac{y_i - \min(y)}{\max(y) - \min(y)}$$

viii. 1. Derece polyfit: Her kenar için $\hat{x} = b\, y_N + c$ modeli uydurulur.

$$\widehat{x_i^L} = b_L\, y_i^N + c_L, \qquad \widehat{x_i^R} = b_R\, y_i^N + c_R$$

ix. Kalan (residual) hesabı yapılır:

$$r_i^L = x_i^L - \widehat{x_i^L}, \qquad r_i^R = x_i^R - \widehat{x_i^R}$$

x. Düzlüklük ölçütü: RMS, her kenarda, doğrusal fit (1. derece) üzerindeki yanal sapmaların (piksel) kök-ortalama-kare değeridir; 'düz' kararı için küçük RMS gereklidir:

$$\mathrm{RMS_L} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i^L)^2} \qquad \mathrm{RMS_R} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_i^R)^2}$$

xi. Eğrilik (2. derece) ve yön bulunur: Kenarın kıvrımlığı için $x \approx a_2\, y_N^2 + b_2\, y_N + c_2$ uyumu alınır; $\mathbf{a_2}$ işareti konkav/konveks yönünü belirtir:

xii. Karar verilir: Polyfit (1. ve 2. derece) ile a2 ve RMS değerlendirildi, düz/içbükey/dışbükey kararı verilir.

$$|a_2| < A_{\text{THR}} \wedge \text{RMS} < \text{RMS}_{\text{THR}} \Rightarrow \text{düz}$$

$$\text{aksi halde işaret ve yöne göre } \{\text{içbükey, dışbükey}\}$$

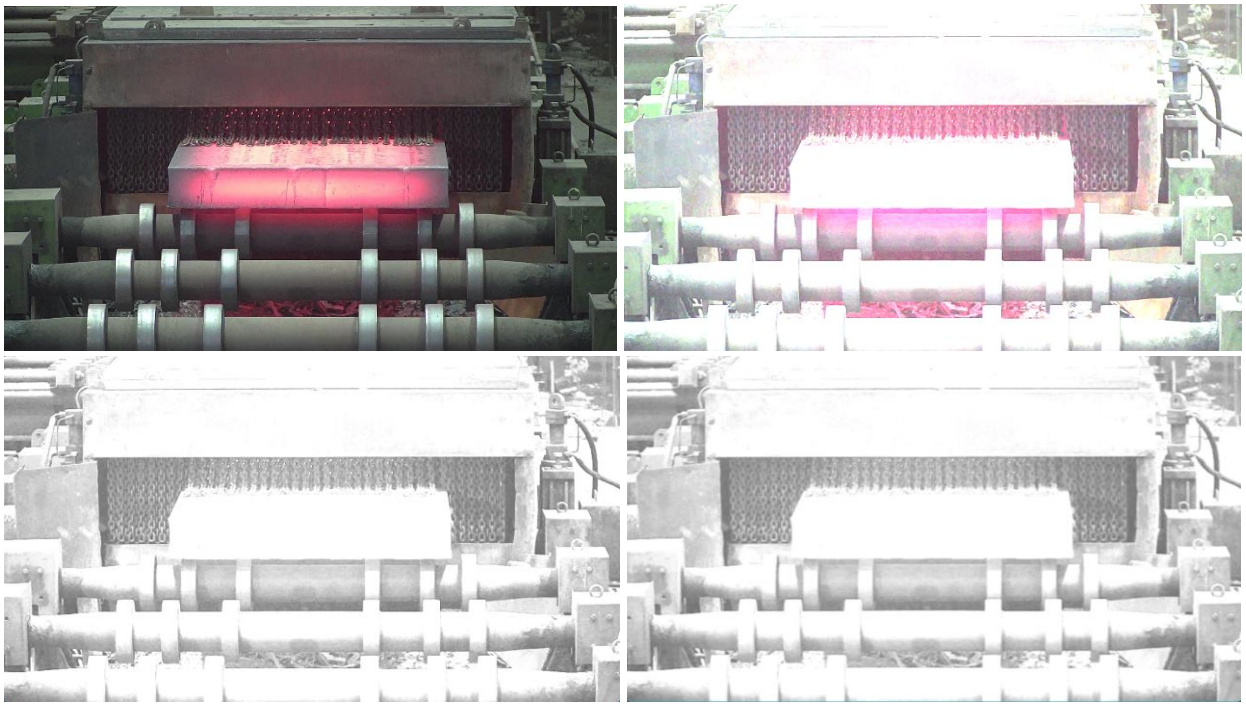
$$\text{Sol kenar: } a_2 < 0 \Rightarrow \text{içbükey}, \quad a_2 > 0 \Rightarrow \text{dışbükey}$$

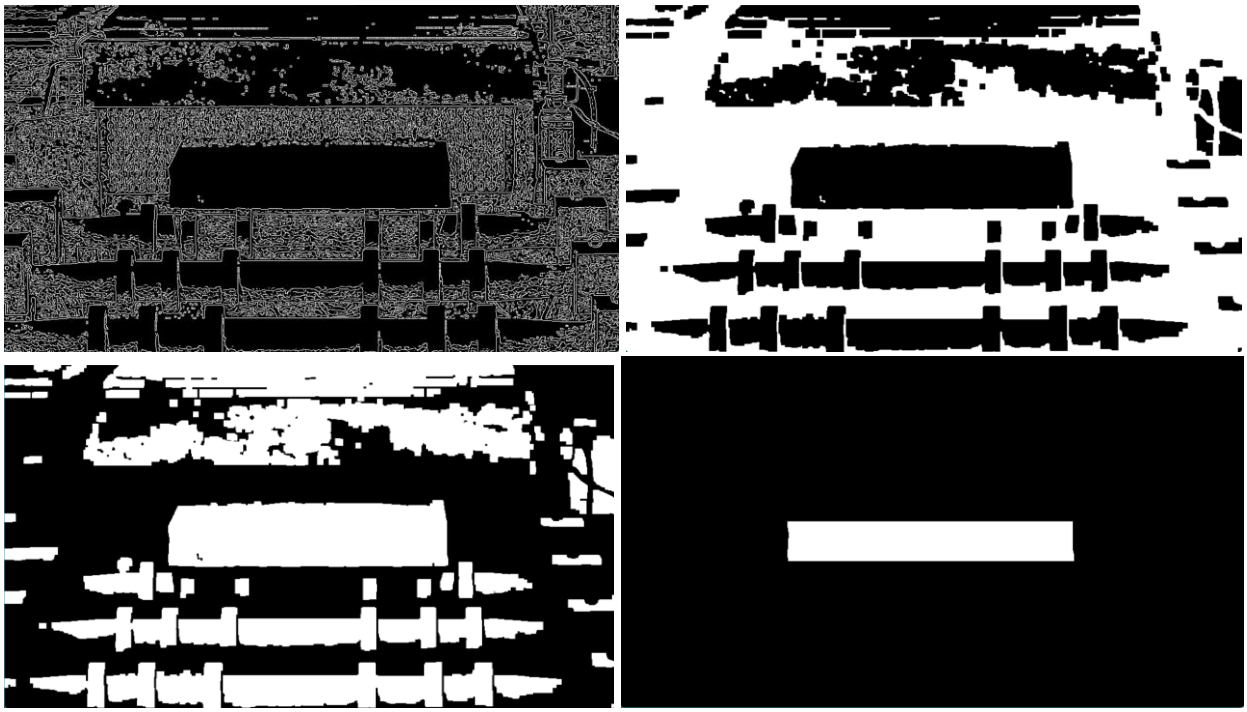
$$\text{Sağ kenar: } a_2 > 0 \Rightarrow \text{içbükey}, \quad a_2 < 0 \Rightarrow \text{dışbükey}$$

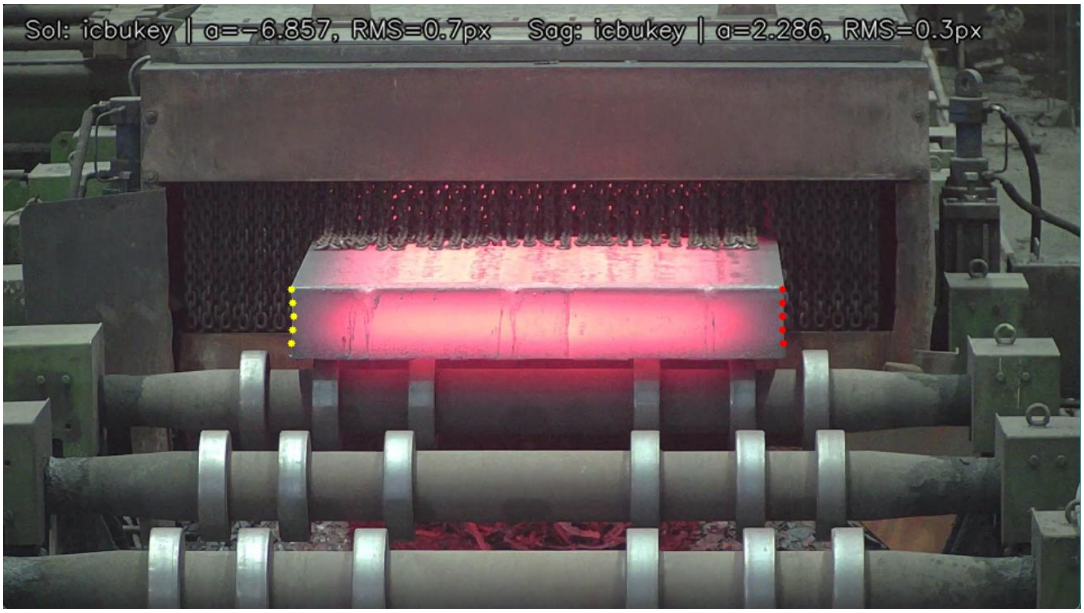
Kenar yönü görüntü koordinat sistemine göre yorumlanır; bu nedenle sol ve sağ kenarda a_2 işaretinin anlamı ters düşer. Bu sebeple sağ ve sol kenar için tanımlamalar yukarıda gösterildiği gibi zıt yapılmıştır.

xiii. Seçilen 5 nokta ve bilgi satırı (a_2 ve RMS) bindirme olarak çizilir; bu, kare bazında denetim sağlar.

Aşağıda görüntünün her işlem sonrasındaki güncel hali listelenmiştir:







Şekil 4- Overlay çıktısı. ROI içinde sol/sağ örnekleme noktalarırms ve düz / içbükey / dışbükey kararı.

3.3. Tetikleme mantığı

OPC üzerinden okunan her yeni zaman değeri görüntü yakalama ve analiz için tetikleyici olarak kullanılır; gerçek sahada bu tetik, slabın sürekli döküm çıkışı konveyöründe tanımlı bir koordinasyon noktasına gelmesine karşılık gelir. Değer 10

saniyeyi aştığında 0'a sarar; bu sarmalar ayrı turlar (epoch) olarak sayılır ve dosya adlarına/CSV'ye işlenir. İstemci tarafında kısa bir bekleme süresi kullanılarak CPU yükü dengelenir ve güncellemeler kaçırılmadan işlenir. Bu yapı, uzun süreli çalışmalarda kararların hem sırayla hem de tur(epoch) bazında izlenebilmesini sağlar.

3.4. Çıktılar ve Kayıt Biçimi

Sistem her tetikte iki tür çıktı üretir. İlki, karar metni ile örnekleme noktalarını içeren bindirmeli görsellerdir; dosya adlarında hem OPC zamanı hem de tur bilgisi yer alır. İkincisi, yapılandırılmış CSV kaydıdır; OPC zamanı, videoda yakalanan gerçek zaman, sol/sağ için eğrilik katsayısı ve RMS ile karar metni sütunlar hâlinde tutulur. Görseller hızlı görsel denetim için, CSV ise istatistiksel analiz ve raporlama için kullanılır.

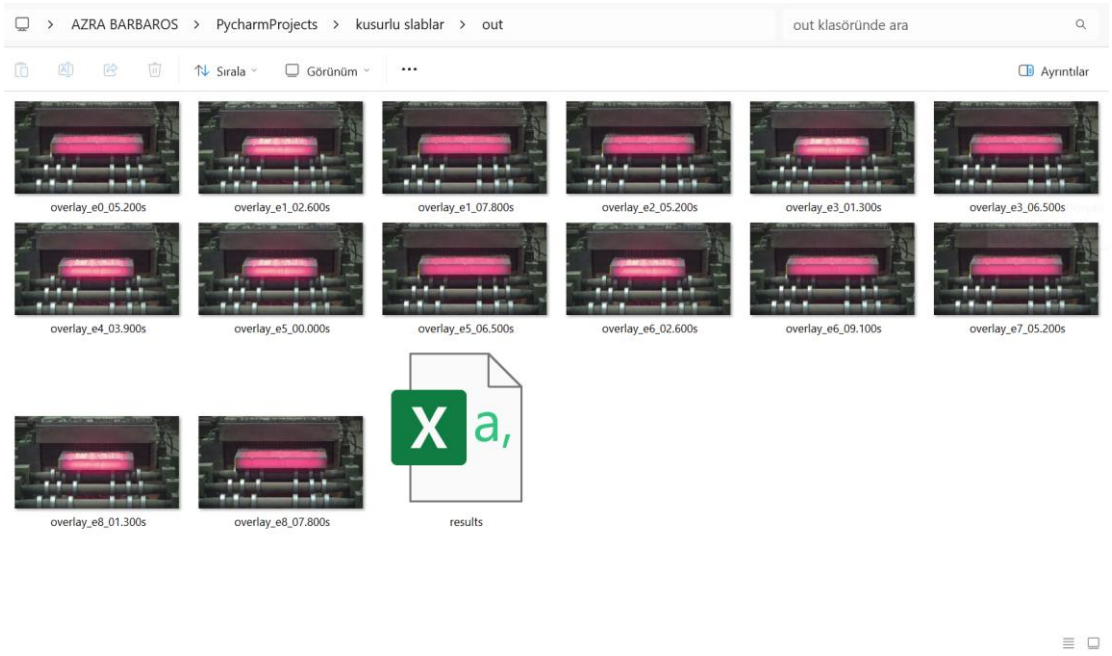
3.5. Entegrasyon – KEPServerEX

Okuma tarafında VideoSecond (Float) etiketi kullanılır; yazma tarafında ResultLeft ve ResultRight (String, RW) etiketleri güncellenir. Access Path boş bırakılır ve UA tarafında değer yazımı doğrudan String Variant olarak yapılır. Yapılandırma, Quick Client ile hem okuma hem yazma yönünde sınanarak doğrulanmıştır.

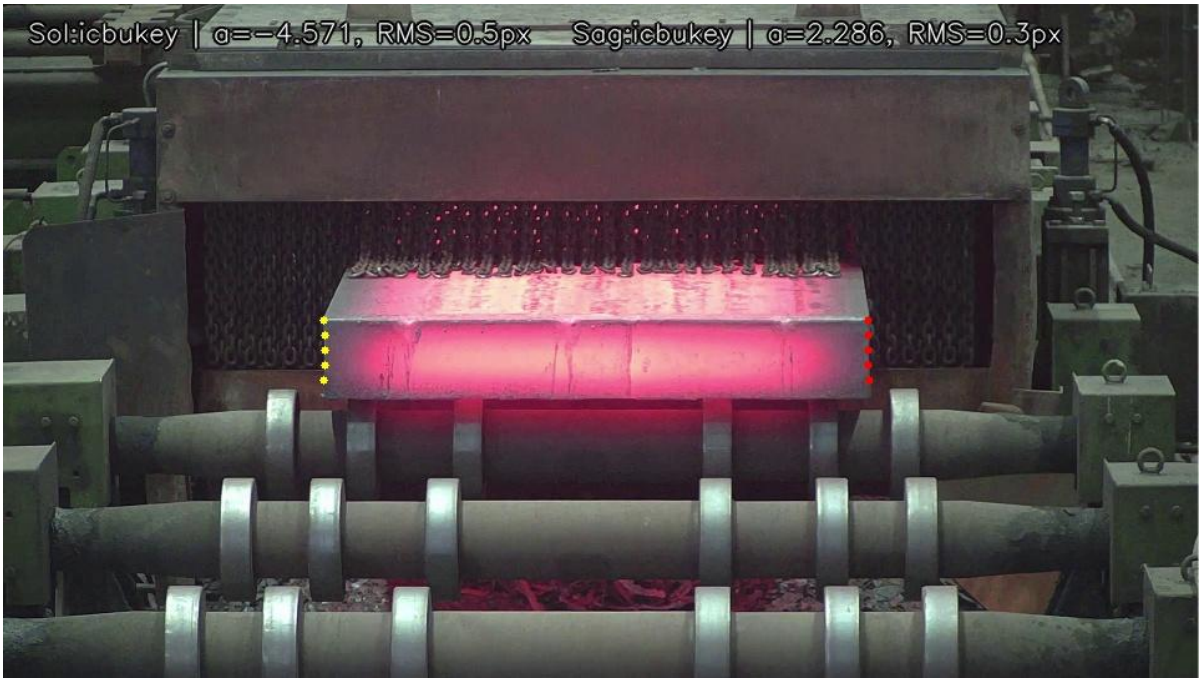
4) Sonuç ve Değerlendirme

Bu çalışmanın çıktıları hem overlay görselleri hem de yapılandırılmış kayıtlar üzerinden birlikte değerlendirilmiştir. Üretilen tüm dosyalar out klasöründe toplanmış, dosya adlarına tur (epoch) ve zaman damgası işlenmiştir (Şekil 5). Örnek overlay karelerinde ROI içinde sol/sağ örnekleme noktaları, polinom uyumu ve "düz / içbükey / dışbükey" karar metni açıkça görülmektedir; bu sayede kare bazında görsel doğrulama yapılabilmektedir (Şekil 6 ve Şekil 7). Sayısal tarafta results.csv, her tetikte OPC zamanı, videoda yakalanan gerçek zaman, sol/sağ a_2 katsayıları ve RMS ile karar etiketlerini satır bazında saklar; böylece eşik ayarlarının ($|a|$ ve RMS) etkisi ve karar kararlılığı izlenebilir (Şekil 8). OPC UA entegrasyonu ise KEPServerEX Quick Client ekranında VideoSecond'ın akışı ve ResultLeft/ResultRight etiketlerine yazılan kararların anlık güncellemeleriyle doğrulanmıştır (Şekil 9).

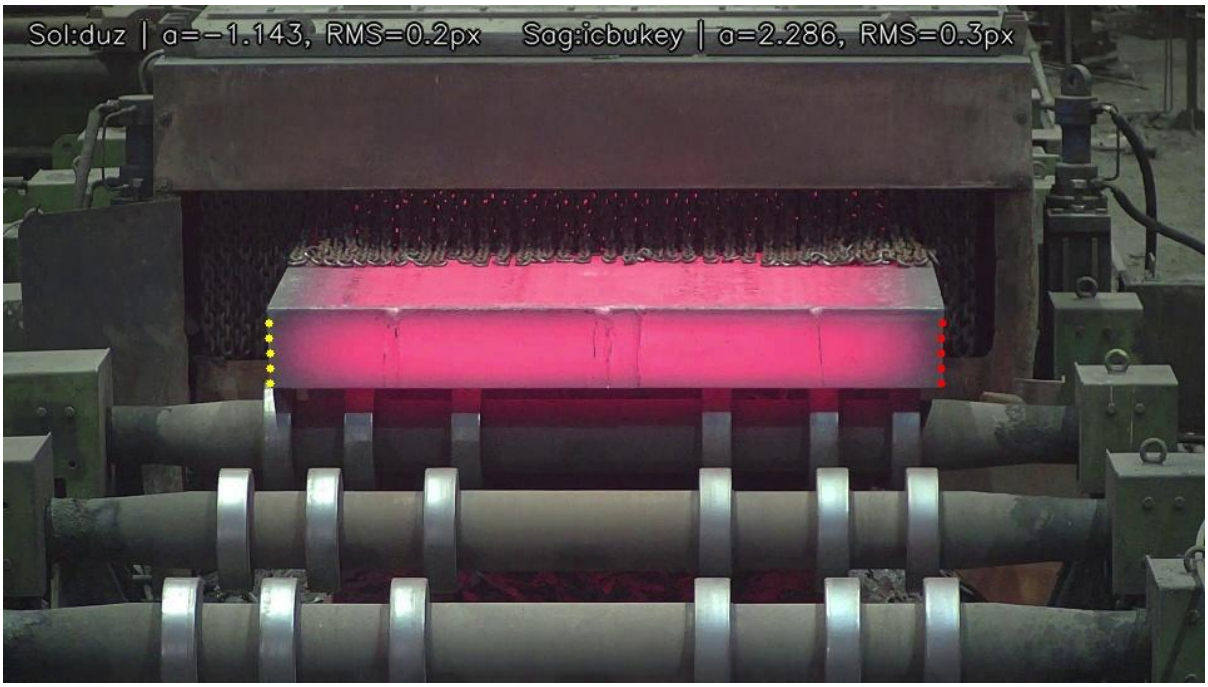
Tüm bu çıktıların birlikte incelenmesi, geliştirilen sistemin hem görsel doğrulama hem de istatistiksel analiz açısından tutarlı ve güvenilir çalıştığını ortaya koymaktadır. Sonuç olarak sistem, sıcak hadde öncesi kalite güvencesi için gerekli olan kenar eğriliği bilgisini sahaya entegre edilmeden dahi doğrulanabilir biçimde üretmektedir.



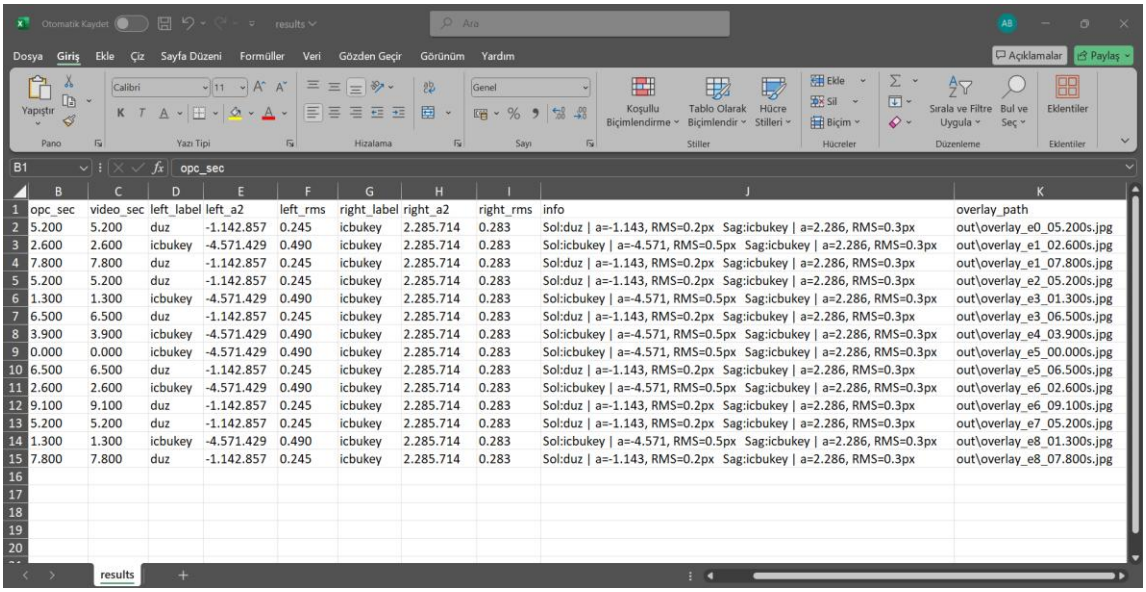
Şekil 5- out klasörü içeriği. Her kare için üretilen overlay görselleri ve toplu sonuçların yer aldığı results.csv dosyası.



Şekil 6- (overlay_e6_02.600s) Overlay örneği (E6, t=02.600 s). ROI içinde sol/sağ örnekleme noktaları karar.

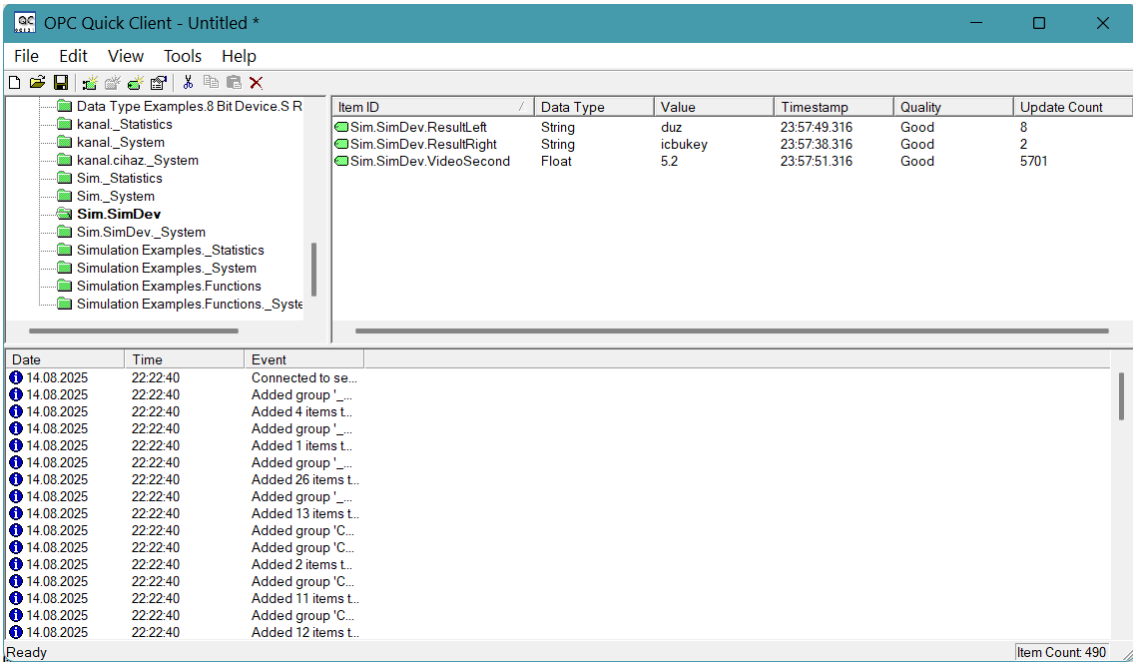


Şekil 7- (overlay_e6_09.100s) Overlay örneği (E6, t=09.100 s). ROI içinde sol/sağ örnekleme noktaları ve karar.



	B	C	D	E	F	G	H	I	J	K
1	opc_sec	video_sec	left_label	left_a2	left_rms	right_label	right_a2	right_rms	info	overlay_path
2	5.200	5.200	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e0_05.200s.jpg
3	2.600	2.600	icbukey	-4.571.429	0.490	icbukey	2.285.714	0.283	Sol:icbukey | a=-4.571, RMS=0.5px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e1_02.600s.jpg
4	7.800	7.800	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e1_07.800s.jpg
5	5.200	5.200	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e2_05.200s.jpg
6	1.300	1.300	icbukey	-4.571.429	0.490	icbukey	2.285.714	0.283	Sol:icbukey | a=-4.571, RMS=0.5px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e3_01.300s.jpg
7	6.500	6.500	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e3_06.500s.jpg
8	3.900	3.900	icbukey	-4.571.429	0.490	icbukey	2.285.714	0.283	Sol:icbukey | a=-4.571, RMS=0.5px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e4_03.900s.jpg
9	0.000	0.000	icbukey	-4.571.429	0.490	icbukey	2.285.714	0.283	Sol:icbukey | a=-4.571, RMS=0.5px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e5_00.000s.jpg
10	6.500	6.500	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e5_06.500s.jpg
11	2.600	2.600	icbukey	-4.571.429	0.490	icbukey	2.285.714	0.283	Sol:icbukey | a=-4.571, RMS=0.5px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e6_02.600s.jpg
12	9.100	9.100	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e6_09.100s.jpg
13	5.200	5.200	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e7_05.200s.jpg
14	1.300	1.300	icbukey	-4.571.429	0.490	icbukey	2.285.714	0.283	Sol:icbukey | a=-4.571, RMS=0.5px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e8_01.300s.jpg
15	7.800	7.800	duz	-1.142.857	0.245	icbukey	2.285.714	0.283	Sol:duz | a=-1.143, RMS=0.2px Sag:icbukey | a=2.286, RMS=0.3px	out\overlay_e8_07.800s.jpg
16										
17										
18										
19										
20										

Şekil 8- results.csv içeriği



Şekil 9- KEPServerEX Quick Client görünümü. Anlık değer, zaman damgası ve kalite alanları üzerinden OPC UA iletişiminin doğrulanması.

5) Ekler

```
1) import os, csv, cv2, asyncio
2) import numpy as np
3) from asyncua import Client, ua
4)
5)
6) VIDEO_PATH = "adsiz.mp4"
7) SAVE_DIR   = "out"
8) CSV_PATH   = os.path.join(SAVE_DIR, "results.csv")
9)
10)     opc_endpoint = "opc.tcp://127.0.0.1:49320"
11)     opc_nodeid = "ns=2;s=Sim.SimDev.VideoSecond" #node'u 2
12)
13)     #kepserverdaki saniye tagının adresi:
14)     # RAMP(100,0,10,1.3) güncelleme periyodu,başlangıç,üstsınır,step(her
   güncellemede kaç artacak)
15)
16)     RESULT_LEFT_NODE  = "ns=2;s=Sim.SimDev.ResultLeft"
17)     RESULT_RIGHT_NODE = "ns=2;s=Sim.SimDev.ResultRight"
18)
19)     CANNY_MIN = 0
```

```
20)     CANNY_MAX = 74
21)     CLOSE_ITER = 5
22)     ALPHA = 3.0
23)     BETA  = 50
24)
25)
26)     ROI_Y0 = 0.47
27)     ROI_Y1 = 0.58
28)     ROI_X0 = 0.15
29)     ROI_X1 = 0.85
30)
31)
32)     A_THR = 1.25
33)     RMS_THR = 5.5
34)
35)     #çok dosya birikmesin diye videodaki yakalama
36)     #MIN_SNAPSHOT_STEP = 0.5
37)
38)
39)     # OPC'ye göre tetikleme:
40)     OPC_DELTA = 0.00   # 0.00 => her yeni değerde kaydet (floating gürültü
   varsa 0.01/0.05 yap)
41)     WRAP_TOL  = 0.2    # 0..10 döngüsünde geriye sarma algısı için
   tolerans
42)     #MAX_EPOCHS = 1     # <<< SADECE 1 TUR (0..10) SONRA DUR
43)
44)
45)     #FONKSİYNLAR:
46)     #-------------------------------------------------------------------
47)
48)     def nothing(x):
49)         pass
50)     #-------------------------------------------------------------------
51)     #def moving_avg(v, k=7):
52)     #    k = max(1, int(k))
53)     #    w = np.ones(k)/k
54)     #    return np.convolve(v, w, mode='same')
55)
56)     #-------------------------------------------------------------------
57)     def decide_curvature(a2, rms, side, a_thr=A_THR, rms_thr=RMS_THR):
58)         # a2 ~ 0 ve RMS küçükse: düz
59)         if abs(a2) < a_thr and rms < rms_thr:
60)             return "duz"
61)         if side == 'left':
62)             # sol kenar: a2>0 => sağa açılır => içbükey
63)             return "disbukey" if a2 > 0 else "icbukey"
64)         else:
65)             # sağ kenar: a2>0 => sağa açılır => dışbükey
66)             return "icbukey" if a2 > 0 else "disbukey"
67)
68)     #-------------------------------------------------------------------
69)     def analyze_frame_with_your_pipeline(img_bgr):
70)
71)         img = cv2.resize(img_bgr, (960, 540))
72)         adjusted = cv2.convertScaleAbs(img, alpha=ALPHA, beta=BETA)
73)         gray = cv2.cvtColor(adjusted, cv2.COLOR_BGR2GRAY)
74)         blur = cv2.GaussianBlur(gray, (5,3), 0)
```

```
75)         edges = cv2.Canny(blur, CANNY_MIN, CANNY_MAX)
76)         edges = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, np.ones((3,3),
   np.uint8), iterations=CLOSE_ITER)
77)
78)         #beyaz çizgi / siyah zemin
79)         mask = cv2.bitwise_not(edges)
80)
81)         #orta bölge filtresi
82)         H, W = mask.shape
83)         y0, y1 = int(H*ROI_Y0), int(H*ROI_Y1)
84)         x0, x1 = int(W*ROI_X0), int(W*ROI_X1)
85)         roi = mask[y0:y1, x0:x1].copy()
86)
87)
88)         roi = cv2.morphologyEx(roi, cv2.MORPH_CLOSE, np.ones((5,5),
   np.uint8), iterations=2)
89)
90)         #en büyük bağlı bileşenin seçimi
91)         num, labels, stats, _ = cv2.connectedComponentsWithStats(roi,
   connectivity=8)
92)         if num > 1:
93)             idx = 1 + np.argmax(stats[1:, cv2.CC_STAT_AREA])
94)             largest = (labels == idx).astype(np.uint8) * 255
95)         else:
96)             largest = roi
97)
98)         mask_big = np.zeros_like(mask)
99)         mask_big[y0:y1, x0:x1] = largest  # tam görüntü boyutuna geri koy
100)
101)        #5 tanw Y satırı(y0 ve y1 dahil)
102)        y_samples = np.linspace(y0, y1, 6, dtype=int)
103)
104)        x_left, x_right, y_used = [], [], []
105)        for y in y_samples:
106)            row = mask_big[y, x0:x1]
107)            xs = np.where(row > 0)[0]  # beyazların x indeksleri (yerel)
108)            if xs.size:
109)                xL = x0 + xs.min()
110)                xR = x0 + xs.max()
111)                y_used.append(y)
112)                x_left.append(int(xL))
113)                x_right.append(int(xR))
114)
115)        y_used = np.array(y_used, dtype=np.float32)
116)        x_left = np.array(x_left, dtype=np.float32)
117)        x_right = np.array(x_right, dtype=np.float32)
118)
119)        overlay = img.copy()
120)
121)        if y_used.size >= 3:  # en az 3 nokta olmalı (2.dereceden)
122)            #sayısal kararlılık için y'yi 0..1 aralığına normalizasyon
123)            yN = (y_used - y_used.min()) / (y_used.max() - y_used.min() +
   1e-9)
124)
125)            #sol kenar
126)            bL, cL = np.polyfit(yN, x_left, 1)
127)            a2L, b2L, c2L = np.polyfit(yN, x_left, 2)
```

```
128)            rmsL = float(np.sqrt(np.mean((x_left - (bL * yN + cL)) ** 2)))
129)
130)            #sağ kenar
131)            bR, cR = np.polyfit(yN, x_right, 1)
132)            a2R, b2R, c2R = np.polyfit(yN, x_right, 2)
133)            rmsR = float(np.sqrt(np.mean((x_right - (bR * yN + cR)) **
   2)))
134)
135)            #karar
136)            decL = decide_curvature(a2L, rmsL, side='left')
137)            decR = decide_curvature(a2R, rmsR, side='right')
138)
139)            #seçilen 5 noktayı çiz
140)            for y, x in zip(y_used.astype(int), x_left.astype(int)):
141)                cv2.circle(overlay, (x, y), 3, (0, 255, 255), -1)  # sol:
   sarı
142)            for y, x in zip(y_used.astype(int), x_right.astype(int)):
143)                cv2.circle(overlay, (x, y), 3, (0, 0, 255), -1)    # sağ:
   kırmızı
144)
145)            info = f"Sol:{decL} | a={a2L:.3f}, RMS={rmsL:.1f}px
   Sag:{decR} | a={a2R:.3f}, RMS={rmsR:.1f}px"
146)            cv2.putText(overlay, info, (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
   0.7, (0, 0, 0), 3, cv2.LINE_AA)
147)            cv2.putText(overlay, info, (20, 30), cv2.FONT_HERSHEY_SIMPLEX,
   0.7, (255, 255, 255), 1, cv2.LINE_AA)
148)
149)            return True, overlay, {"left": {"label": decL, "a2":
   float(a2L), "rms": float(rmsL)},
150)                                   "right":{"label": decR, "a2":
   float(a2R), "rms": float(rmsR)},
151)                                   "info": info}
152)        else:
153)            cv2.putText(overlay, "5 satir icin yeterli piksel yok", (20,
   30),
154)                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
155)            return False, overlay, {"info":"yetersiz_nokta"}
156)
157)    #-------------------------------------------------------------------
158)    def read_video_meta(path):
159)        cap = cv2.VideoCapture(path)
160)        if not cap.isOpened():
161)            raise RuntimeError(f"Video acilamadi: {path}")
162)        fps = cap.get(cv2.CAP_PROP_FPS) or 30.0
163)        total = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
164)        dur = total / fps
165)        cap.release()
166)        return fps, total, dur
167)
168)    #-------------------------------------------------------------------
169)    def snap_frame_at_second(path, sec):
170)        cap = cv2.VideoCapture(path)
171)        if not cap.isOpened():
172)            raise RuntimeError("Video acilamadi")
173)        fps = cap.get(cv2.CAP_PROP_FPS) or 30.0
174)        total = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
175)        dur = total / fps
```

```
176)
177)        s = max(0.0, min(float(sec) % dur, dur - 1e-3))  # 0..duration
178)        cap.set(cv2.CAP_PROP_POS_MSEC, s * 1000.0)
179)        ok, frame = cap.read()
180)        if not ok:
181)            target_idx = int(round(s * fps))
182)            cap.set(cv2.CAP_PROP_POS_FRAMES, max(0, min(target_idx, total
    - 1)))
183)            ok, frame = cap.read()
184)            if not ok:
185)                cap.release()
186)                raise RuntimeError("Kare alinamadi")
187)        cap.release()
188)        return frame, s, dur
189)
190)    #KAYIT----------------------------------------------------------------
191)    def append_csv(path, row):
192)        os.makedirs(os.path.dirname(path), exist_ok=True)
193)        write_header = not os.path.exists(path)
194)        with open(path, "a", newline="", encoding="utf-8") as f:
195)            w = csv.DictWriter(f, fieldnames=row.keys())
196)            if write_header: w.writeheader()
197)            w.writerow(row)
198)
199)    #ANA AKIŞ (polling)---------------------------------------------------
200)    async def main():
201)        os.makedirs(SAVE_DIR, exist_ok=True)
202)        fps, total, dur = read_video_meta(VIDEO_PATH)
203)        print(f"Video: {dur:.3f} sn, {fps:.2f} fps, {total} frame")
204)
205)        last_written = -1e9  # snapshot aralığı kontrolü
206)        last_opc = None  # en son işlenen OPC saniyesi
207)        prev_opc = None  # sargı (wrap) algısı için
208)        epoch = 0  # 0..10 döngüsü sayacı (her başa sarışta ++)
209)
210)        async with Client(url=opc_endpoint) as client:
211)            node = client.get_node(opc_nodeid)
212)            node_left = client.get_node(RESULT_LEFT_NODE)
213)            node_right = client.get_node(RESULT_RIGHT_NODE)
214)            while True:
215)                val = await node.read_value()
216)                try:
217)                    sec = float(val)
218)                except:
219)                    await asyncio.sleep(0.2)#CPU tasarrufu için iyi ama
    server hızlıysa 0.05 filan da olabilir
220)                    continue
221)
222)                # 0..10 döngüsünde başa sarma(prev_opc > curr_opc)
223)                if prev_opc is not None and (sec + WRAP_TOL) < prev_opc:
224)                    epoch += 1
225)                prev_opc = sec
226)
227)                # OPC güdümlü tetik: her yeni değer (veya en az OPC_DELTA
    kadar fark)
228)                if (last_opc is None) or (abs(sec - last_opc) >=
    OPC_DELTA):
```

```
229)                     last_opc = sec
230)
231)                     frame, used_sec, _ = snap_frame_at_second(VIDEO_PATH,
   sec)
232)
233)                     ok, overlay, res =
   analyze_frame_with_your_pipeline(frame)
234)
235)                     # dosya adı ve CSV'de hem OPC saniyesi hem video
   saniyesi dursun
236)                     jpg = os.path.join(SAVE_DIR,
   f"overlay_e{epoch}_{sec:06.3f}s.jpg")
237)                     cv2.imwrite(jpg, overlay)
238)
239)                     row = {
240)                         "epoch": epoch,  # kaçıncı 0..10 turu
241)                         "opc_sec": f"{sec:.3f}",  # server'ın gönderdiği
   ham saniye
242)                         "video_sec": f"{used_sec:.3f}",  # videoda
   yakalanan gerçek saniye
243)                         "left_label": (res["left"]["label"] if ok else
   ""),
244)                         "left_a2": (f'{res["left"]["a2"]:.6f}' if ok else
   ""),
245)                         "left_rms": (f'{res["left"]["rms"]:.3f}' if ok
   else ""),
246)                         "right_label": (res["right"]["label"] if ok else
   ""),
247)                         "right_a2": (f'{res["right"]["a2"]:.6f}' if ok
   else ""),
248)                         "right_rms": (f'{res["right"]["rms"]:.3f}' if ok
   else ""),
249)                         "info": res["info"],
250)                         "overlay_path": jpg
251)                     }
252)                     append_csv(CSV_PATH, row)
253)                     print(f"E{epoch} OPC={sec:05.2f} ->
   video={used_sec:05.2f} | {res['info']} | {jpg}")
254)
255)                     # (opsiyonel) sonucu OPC'ye yazıyorsan:
256)                     # await
   node_left.write_value(ua.Variant(res["left"]["label"] if ok else "NA",
   ua.VariantType.String))
257)                     # await
   node_right.write_value(ua.Variant(res["right"]["label"] if ok else "NA",
   ua.VariantType.String))
258)
259)                     #kepserverdeki string taglarına yazacak
260)                     left_str = res["left"]["label"] if ok else "NA"
261)                     right_str = res["right"]["label"] if ok else "NA"
262)
263)                     try:
264)                         dvL = ua.DataValue(ua.Variant(left_str,
   ua.VariantType.String))
265)                         dvR = ua.DataValue(ua.Variant(right_str,
   ua.VariantType.String))
266)
```

```
267)                         # StatusCode/SourceTimestamp/ServerTimestamp gibi
   alanları SET ETME!
268)                         await
   node_left.write_attribute(ua.AttributeIds.Value, dvL)
269)                         await
   node_right.write_attribute(ua.AttributeIds.Value, dvR)
270)                     except Exception as e:
271)                         print("OPC write failed (DataValue):", e)
272)
273)                 await asyncio.sleep(0.2)  # 200 ms polling
274)
275)     #Kodu çalıştırma-----------------------------------------------------------
276)     if __name__ == "__main__":
277)         try:
278)             asyncio.run(main())
279)         except KeyboardInterrupt:
280)             print("bitti")
```